This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2009 proceedings.

TCP/IP Timing Channels: Theory to Implementation

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

Ness Shroff
Departments of ECE and CSE
The Ohio State University
Columbus, OH 43210
shroff@ece.osu.edu

***Abstract*—There has been significant recent interest in covert communication using timing channels. In network timing channels, information is leaked by controlling the time between transmissions of consecutive packets. Our work focuses on network timing channels and provides two main contributions. The first is to quantify the threat posed by covert network timing channels. The other is to use timing channels to communicate at a low data rate without being detected.**

In this paper, we design and implement a covert TCP/IP timing channel. We are able to quantify the achievable data rate (or leak rate) of such a covert channel. Moreover, we show that by sacrificing data rate, the traffic patterns of the covert timing channel can be made computationally indistinguishable from that of normal traffic, which makes detecting such communication virtually impossible. We demonstrate the efficacy of our solution by showing significant performance gains in terms of both data rate and covertness over the state-of-the-art.

I. Introduction

The Orange Book [1] defines a covert channel to be *"any communication channel that can be exploited by a process to transfer information in a manner that violates the system's security policy."* A covert timing channel is a type of covert channel in which sensitive information is transmitted by the timing of events. In a multi-level security (MLS) system, covert timing channels can be used by a HIGH process to leak classified information to a LOW process. In a networked environment, it can be used by a program that has access to sensitive information to leak the information through packet inter-transmission times.

Designing and implementing timing channels over a shared network between two distant computers is challenging. Network timing channels are inherently "noisy" due to the delay and jitter in networks, which distort the timing information from the sender when it reaches the receiver. In [2], the authors designed and implemented an IP covert timing channel using an on-off coding scheme, where the reception or absence of a packet within a time interval signals bit 1 or bit 0, respectively. This timing channel achieves a data rate of 16.67 bits/sec between two computers located at two universities with an average round trip time of 31.5 ms.

In TCP/IP networks, end-to-end delays are much larger than the jitter. Information theoretic research shows that the Shannon capacity of timing channels with no jitter is infinite, assuming infinite precision of the system clock ([3], [4], [5], [6], [7], [8]), and the capacity can be made very large if the jitter of the underlying channel is very small [10]. Motivated by these theoretical results, we are interested in designing a TCP/IP covert timing channel that significantly improves the current state-of-the-art data rate [2]. Additionally, our second goal is to design a computationally non-detectable timing channel scheme, assuming that the packet inter-transmission time of regular traffic is independently and identically distributed (*i.i.d.*), but the distribution of the inter-transmission times could be general (e.g., Pareto).

In our design, we use packet *inter*-transmission times (denoted as T_k) to convey information. Figure 1 shows a high-level view of our design. A malicious process on the sender side manipulates the inter-transmission times and another malicious process either at the receiver or en route to the receiver can decode the privileged information by observing the inter-reception times. We encode L-bit binary strings in a sequence of n packet inter-transmission times $T_1, T_2, \cdots, T_n$. We call it the "*L-bits to n-packets*" scheme. These n packets are transmitted in *variable* length time intervals. The receiver will then map the n packet inter-reception times $R_1, R_2, \cdots, R_n$ back to an L-bit binary string according to the code book.

We make two novel contributions in this paper (both are also distinct from our prior work [10]). First, we provide a systematic solution of selecting the values of L and n for the *L-bits to n-packets* scheme, so that the data rate of our scheme is near optimal. We implement a TCP/IP timing channel with our *L-bits to n-packets* scheme, and conduct extensive experiments on the PlanetLab environment [9] with five pairs of computers distributed worldwide to show the effect of differing delay and jitter. We demonstrate significant performance improvement (2 to 5 times the covert timing channel data rate) of our scheme over the state-of-the-art [2].

Our second contribution is to systematically develop a computationally non-detectable timing channel scheme, assuming the packet intertransmission time is *i.i.d.* Our design is based on the security of the *cryptographically secure pseudo random number generators* (CSPRNG). The packet inter-transmission times (T_k's) from the proposed timing channel are devised to mimic any legitimate traffic with *i.i.d.* packet inter-transmission time, e.g., the *i.i.d.* Pareto distribution for telent traffic [20]. This allows two parties to communicate at a low data rate (e.g., 5 bits/sec) in a hostile environment such as in battlefield or law enforcement settings while avoiding detection. The proposed non-detectable scheme is also implemented, and experiments are conducted on PlanetLab. The

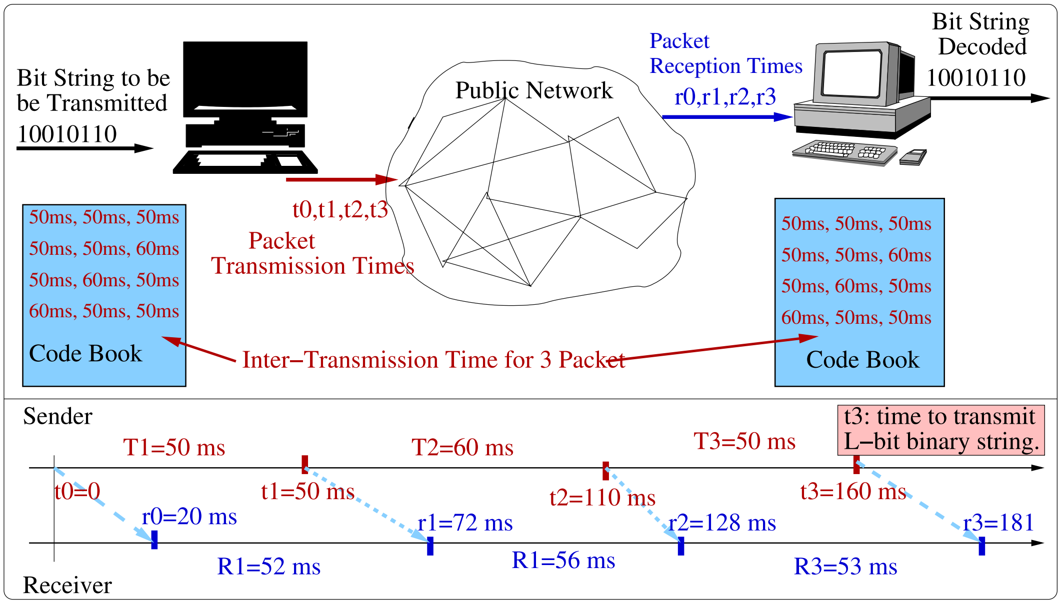


Fig. 1. High Level Design Diagram of Covert Timing Channel

similarity of the traffic patterns of our scheme and the telnet traffic is verified.

The remainder of our paper is organized as follows: In Section II, we review related work. In Section III, we present our system level design and the proposed *L-bits to n-packets* scheme, with discussions on the trade-offs between the data rate and the complexity of our scheme. In Section IV, we describe our implementation of covert TCP/IP timing channels and our experimental results. In Section V, we show how to construct a timing channel scheme that is computationally not detectable. We conclude with discussion and future research directions in Section VI.

II. Related Work

The best achievable data rate of a covert timing channel measures the severity of its threat, and can be obtained by estimating the information theoretic channel capacity [3], [5], [10], [11]. To reduce the throughput of covert timing channels, several methods such as timing channel jammers [6], fuzzy times [12] and network pump [13], have been proposed. Another approach to defend against the usage of covert timing channels is to detect the presence of such usage [2], [15], [16].

In [2], the authors illustrated the threat from IP covert timing channels using a software implementation of covert timing channels over TCP/IP networks. They also developed a detection mechanism for their IP covert timing channels by identifying the regularity of the inter-transmission times. In their design, the sender and receiver agree upon an interval length, say w ms. To signal bit value ‘1’, the sender transmitted a packet in the middle of the time interval; to signal ‘0’, the sender remained silent during that interval. Their scheme achieved a rate of 16.67 b/s with 2% character decoding error (w = 60 ms) in an experimental setting where the average RTT between the two hosts is 31.5 ms. Their use of a simple coding scheme, basically an on-off scheme, limits the data rate achievable for the timing channel. Their scheme requires time synchronization between the sender and the receiver in order to correctly decode a message. A constant shift in delay and jitter may have a cascade effect that may cause subsequent bits to be decoded incorrectly.

In [17], 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. The authors utilized the interactive session and added different delays to the timing sequence of the keystrokes to signal 1 or 0. Their binary encoding scheme allows one bit of information to be transmitted per keypress, and the rate of their timing channel scheme is limited by the user’s typing speed. They then used a “4-bit to 1-keypress” encoding scheme to improve the performance. This method uses 16 different delay values to encode the 16 distinct 4-bit binary strings. The timing channel rate being tied to human input is very low, and it is detectable by sophisticated detection algorithms such as [16].

In our proposed scheme, we use packet inter-transmission time to convey information, not just the presence or absence of a packet in a fixed time interval. Our method eliminated the need for absolute time synchronization between the sender and the receiver, and it shares some similarities with the *sparse time base* used for clock synchronizations in distributed systems in [18].

We propose a more general *L-bits to n-packets* scheme that maps binary strings of length L to multiple packet inter-transmission times of size n, which includes both the *on-off* scheme [2] and the keyboard jitter bugs as special cases. We further provide methods for selecting the values of parameters L and n to get higher data rates. Finally, we describe our design of a timing channel scheme that mimics a class of normal traffic to avoid detection.

III. Covert Timing Channel Design

A. System Level Model and Design

Our approach can be applied to a wide variety of communication paradigms to transmit covert information. However, for the sake of illustration, we describe here a high level view (illustrated in Figure 1) for communication with TCP/IP. The sender and receiver reside on two distant computers with several routers in between. The sender has access to sensitive information and wants to leak this information to the receiver using the timing of packet transmissions.

We use t_k and r_k to denote the times that the k^{th} packet is transmitted and received, respectively. We use D_k to denote the end-to-end delay of the k^{th} packet, so that $r_k = t_k + D_k$. The delay D_k include the delays from the TCP/IP network and the processing delays on both computers running the timing channel software. It can be expressed as $D_k = D + \epsilon_k$, where D is a constant that represents the average delay, and ϵ_k is a random variable that represents the jitter. In TCP/IP networks, jitter is typically bounded (e.g., $-\epsilon_{max} < \epsilon_k < \epsilon_{max}$), but may not be *i.i.d.*.

The packet inter-transmission (denoted as T_k) between the $(k-1)^{st}$ packet and the k^{th} packet can be expressed as $T_k = t_k - t_{k-1}$. Likewise, the packet inter-reception time (denoted as R_k) between the $(k-1)^{st}$packet and the k^{th} packet can be expressed as $R_k = r_k - r_{k-1}$. Thus,

$$R_k = T_k + (\epsilon_k - \epsilon_{k-1}) \tag{1}$$

Equation (1) models the timing channel, with T_k being the input to the channel and R_k being the noisy output.

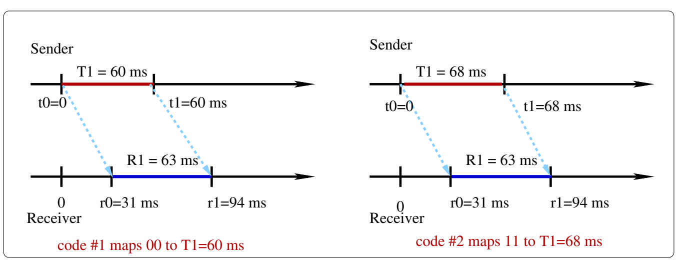


Fig. 2. Identical R_1 for two distinct bit strings resulting in decoding error for the timing channel

Let $T_1^{(1)}$ and $T_1^{(2)}$ be any two inter-transmission times representing two distinct binary strings. We require that $|T_1^{(1)} - T_1^{(2)}|$ be large enough so that the corresponding inter-reception times $R_1^{(1)}$ and $R_1^{(2)}$ are *always* distinguishable even with the noisy component $(\epsilon_k - \epsilon_{k-1})$. Figure 2 illustrates a scenario when $|T_1^{(1)} - T_1^{(2)}|$ is too small. In this example, $D = 30$ ms and $|\epsilon_k| < 5$ ms for all k. Suppose we encode "00" as $T_1^{(1)} = 60$ ms, and "11" as $T_1^{(2)} = 68$ ms. As shown in the figure, the inter-reception times $R_1^{(1)}$ and $R_1^{(2)}$ for "00" and "11" can be the same, for which it is impossible for the receiver to decide if "00" or "11" was sent.

In our design, we use the parameter δ to denote this minimum difference of $T_1^{(1)}$ and $T_1^{(2)}$, and $\delta > 4\epsilon_{max}$ ensures no confusion for any values of ϵ_k and ϵ_{k-1}. The reason is that when $|\epsilon_k| < \epsilon_{max}$, it follows directly from equation (1) that

$$T_1^{(1)} - 2\epsilon_{max} < R_1^{(1)} < T_1^{(1)} + 2\epsilon_{max} \tag{2}$$
$$T_1^{(2)} - 2\epsilon_{max} < R_1^{(2)} < T_1^{(2)} + 2\epsilon_{max} \tag{3}$$

If $|T_1^{(1)} - T_1^{(2)}| > 4\epsilon_{max}$, then (2) and (3) guarantees that $R_1^{(1)} \neq R_1^{(2)}$.

Another parameter in our code design is Δ, the minimum value for T_k (i.e., $T_k \geq \Delta$). The intuition for imposing a minimum time Δ between the transmissions of any two consecutive packets is to avoid loss of timing information. If two packets are transmitted too close to each other, they may be queued at the computers running the timing channel software or on the intermediate routers, and queueing could destroy the timing information [3].

Even though a better designed timing channel can achieve a higher data rate, we must also keep in mind that the data rate cannot be arbitrarily large since it is upper bounded by the channel's Shannon capacity, the maximum possible data rate for two parties to communicate reliably. In general, the Shannon capacity of timing channels is not known. However, we show in [10] that among all *i.i.d.* continuous and bounded jitter distributions in $(-\epsilon_{max}, \epsilon_{max})$, the Shannon capacity is the smallest when the jitter distribution is uniform in that interval. In other words, the uniform jitter distribution represents the worst case scenario in terms of channel capacity. A special map of the bit-string to packet inter-transmission time, termed the geometric code is proposed in [10]. We have shown in [10] that the rate of the geometric code comes very close to the true capacity of the timing channel with uniform jitter distribution. It is worth noting that the geometric code is universal that works with all bounded jitter even when the underlying unknown distribution is non *i.i.d.*.

We next describe the geometric code and present a simple realization of the geometric code by the *L-bits to n-packets* scheme. We then evaluate the performance of our *L-bits to n-packets* scheme.

B. Geometric Codes

The family of geometric codes are those codes with T_i to be $i.i.d$ geometric random variables with *probability mass function* :

$$P[T_i = \Delta + k \cdot \delta] = p(1-p)^k, \quad k = 0, 1, 2, \cdots$$

We have shown in [10, Lemma 1] that the data rate of such a geometric code is

$$R(p) = \frac{H(p)}{\Delta \cdot p + \delta \cdot (1-p)}$$

where $H(p) = -p\log_2(p) - (1-p)\log_2(1-p)$, and the achievable data rate for any geometric code is:

$$R^* = \max_{0<p<1} [H(p)/(\Delta \cdot p + \delta \cdot (1-p))].$$

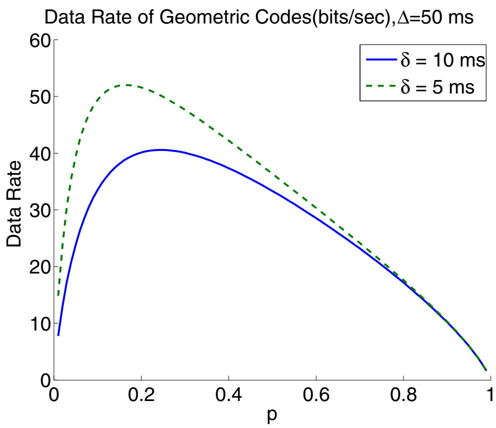


Fig. 3. $R(p)$ for geometric codes.

Figure 3 shows the data rate $R(p)$ as a function of p for two geometric codes with system parameters (Δ, δ) set to $(50, 10)$ ms and $(50, 5)$ ms. As shown, when $\delta = 5$ ms, $R(p)$ attains its maximum value $R^* = 52$ bits/sec. Likewise, when $\delta = 10$ ms, $R(p)$ attains its maximum value $R^* = 40.56$ bits/sec. It is not surprising that with fixed Δ, the achievable rate R^* is higher when δ is smaller – the smaller δ is, the shorter time it takes to transmit all packets.

C. L-bits to n-packets Scheme

Our design of the *L-bits to n-packets* scheme is based on insights from geometric codes. In this scheme, each of the *L-bit* binary strings is mapped to a sequence of packet inter-transmission times $(T_1, T_2, \cdots, T_n)$. In our basic scheme, T_i takes values only from the set $E = \{T : \Delta + k \cdot \delta, \quad k = 0, 1, \cdots\}$.

To illustrate our design, we first give examples of a "*2-bit to 1-packet*" scheme and a "*4-bit to 1-packet*" scheme with $\Delta = 60$ ms and $\delta = 20$ ms. (Actually, for the first L bit string, $(n+1)$ packets are needed including the starting packet. But for subsequent L-bit strings, only n packets are needed.) A "*2-bit to 1-packet*" scheme maps bit string "10" in one inter-transmission time $T_1 = 60$ ms. Likewise, it maps bit strings "01", "11", and "00" to T_1=80 ms, 100 ms, and 120 ms, respectively. On average, it takes 90 ms to transmit 2 bits, assuming each bit string is equally likely. So, the data rate is $\frac{2}{0.09} \approx 22$ bits/sec.

Now consider a "*4-bits to 1-packet*" encoding scheme. A total of 16 different values for the inter-transmission times T_1 are needed to represent all the 4-bit binary strings. We can use the following 16 values for the inter-transmission times T_1: 60, 80, 100, $\cdots$, 340, and 360 ms. On average, it takes 210 ms to transmit 4 bits, and the data rate is $\frac{4}{0.21} \approx 19$ bits/sec.

In these examples, the *2-bits to 1-packet* scheme outperforms the *4-bits to 1-packet* scheme in terms of data rate. It may appear from this example that the data rate of the timing channel monotonically decreases with increasing L. However, the interesting fact our investigation reveals is that the rate is not monotonic with L.

One design challenge is to determine the values for L and n, so that our timing channel achieves a near optimal throughput – close to the data rate of the corresponding geometric code. Thus, given a fixed total packets transmission time t_n ($t_n = \sum_{i=1}^{n} T_i$), we would like to transmit the longest bit strings possible.

To aid our analysis, we introduce another n-dimensional vector $\mathbf{k} = (k_1, k_2, ..., k_n)$ to represent $\mathbf{T} = (T_1, T_2, \cdots, T_n)$, for $T_i = \Delta + k_i \cdot \delta$. We consider a special *L-bits to n-packets* scheme, called an (n, K)*-code*, that satisfies $\sum_{i=1}^{n} k_i \leq K$. Using an (n, K)-code, $t_n \leq n \cdot \Delta + K \cdot \delta$. We have shown in [19], that the maximum number of available codewords in an (n, K)-code is $\binom{n+K}{K}$. Therefore, the maximum length of a bit strings that can be mapped to an (n, K)-code is

$$L = \lfloor \log_2 \binom{n+K}{K} \rfloor. \quad (4)$$

The data rate $R(n, K)$ of an (n, K)-code with system parameters (Δ, δ) is thus approximately

$$R(n, K) \approx \frac{\log_2 \binom{n+K}{K}}{n \cdot \Delta + \frac{n}{n+1} \cdot K \cdot \delta} \quad \text{bits/sec.} \quad (5)$$

A plot of $R(n, K)$ as a function of K is shown in Figure 4, for two values of n. In this figure, $(\Delta, \delta) = (50, 10)$ ms, and $n = 3, 5$. As shown, for a fixed value of n, the data rate $R(n, K)$ will initially increase as K increases till it reaches its peak, and it then decreases as K increases. The intuition is that with increasing K, the total number of codewords $\binom{n+K}{K}$ is increasing. Meanwhile, t_n, the time it takes to transmit all n packets, will also increase with K. Initially, the gain in the total number of codewords outpaces the increase in t_n, and we see an increase of the data rate. After a certain point, increase in t_n outpaces the gain in the total number of codes, and we see a decrease of the data rate.

For a fixed value of n, the highest data rate using (n, K)-codes with system parameters (Δ, δ) is approximately

$$R^*(n) \approx \max_{K \geq 0} \frac{\log_2 \binom{n+K}{K}}{(n \cdot \Delta + \frac{n}{n+1} \cdot K \cdot \delta)} \quad \text{bits/sec.} \quad (6)$$

In the $n = 3$ case, the $(3, K)$-code achieves its highest data rate $R^*(3) = 36.96$ bits/sec when $K = 13$. The optimal value of L can be calculated using formula (4). The total number of codewords is $\binom{16}{13}$ = 560, and $L = 9$. Thus, when $n = 3$, a *9-bits to 3-packets* gives us the best data rate. Likewise, when $n = 5$, a *15-bits to 5-packets* scheme yields the highest data rate of 37.73 bits/sec.

Figure 5 shows the optimal data rate $R^*(n)$ as a function of L for the same system parameter $(\Delta, \delta) = (50, 10)$ (ms), along with the rate of the corresponding geometric code. In general, $R^*(n)$ increases as L increases. When L is large, $R^*(n)$ is very close to the geometric code rate. However, the complexity of the codes also increases as L increases. As shown in this figure, in order to gain a small amount of data rate $R^*(n)$, we must increase L drastically. For instance, using a *9-bits to 3-packets* scheme yields a rate of 37 bits/sec, while to achieve a

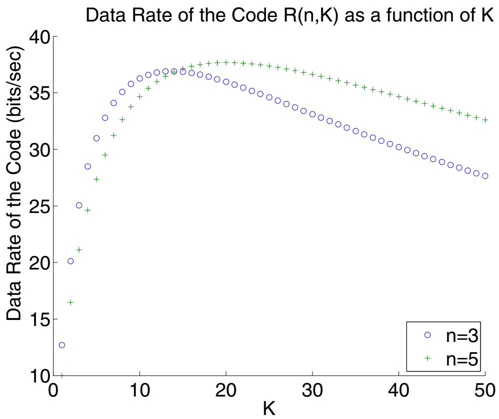


Fig. 4. $R(n, K)$ for $\Delta = 50$ ms, $\delta = 10$ ms.

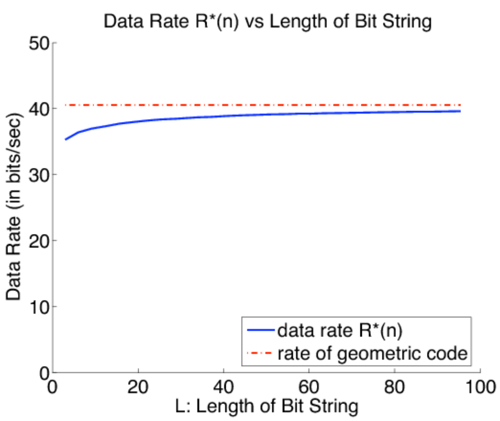


Fig. 5. $R^*(n)$ for $\Delta = 50$ ms, $\delta = 10$ ms.

data rate of 39 bits/sec, we need to use a *66-bits to 32-packets* scheme. The latter is much more expensive in terms of storage for the code book and processing for encoding and decoding, since 2^{66} codewords will have to be stored and searched for. However it only offers very little gain in data rate (2 bits/sec).

IV. Experimental Results

Based on our design, we have developed a covert timing channel software running over TCP/IP networks. Our software is implemented in Java, consisting of a server program and a client program that act as the sender and the receiver, respectively. The sender controls the TCP packet inter-transmission time by using $sleep(T)$, where T is the desired time (in milliseconds) between two packets being transmitted. The accuracy of the $sleep(T)$ function on our system is 1 ms. The receiver passively collects the TCP packet reception times, and uses the shared code book to decode the message. It is a one-way channel in that the sender does not receive feedback from the receiver regarding when the packet is received or whether it is decoded correctly. This limits the performance of the timing channel but increases the difficulty of detection. In our implementation, we choose an *8-bits to 3-packets* scheme for simplicity and efficiency.

As mentioned in our design, our timing channel does not require time synchronization between the sender and the receiver, which makes it attractive for an open network like the Internet. Moreover, the errors occurring earlier will not affect the decoding capability of messages sent later because of independent decoding of each L-bit string. This is in contrast to [2], where a packet delay will cause subsequent bits to be erroneously decoded.

We conducted our experiments using the PlanetLab environment. We ran our covert timing channel software on five pairs of computers. The senders are hosts at Purdue University, and the receivers are PlanetLab nodes located at Beijing Tsinghua University, Technical University of Madrid, University of Zurich, Stanford University and Princeton University. These five pairs are chosen to represent a wide range of Round Trip Times between the senders and receivers. At the receiver side, we use the packets captured by *tcpdump* to decode the timing channel message.

In a single experiment, the sender leaks the information obtained from a text file of 1336 ascii characters to the receiver via our covert timing channel. A set of experiments consists of 10 such experiments with system parameters (Δ, δ): $\Delta = 10, 20, 30, 40, 50$ (ms) and $\delta = 5, 10$ (ms). We ran the set of 10 experiments daily between these five pairs of sender and receiver during morning hours (EST) for 10 days.

Table I summarizes our results of these experiments. In the table, we provide the average and the standard deviation of the character decoding error of our experiments. If one decoded character does not match the transmitted character, it is counted as one decoding error. In our *8-bits to 3-packets* scheme, if one of the three packet inter-reception times deviates too much from the corresponding inter-transmission time, it will result in one character decoding error (1 character is represented by 8 bits). Thus, a 1% of packet inter-reception time error could result in a 3% character decoding error. In addition to the results on error rates, the actual data rate for all the system parameters (Δ, δ), and the average RTT time between the pair just before our experiments are also shown in this Table. Two general trends are observable. As Δ or δ decreases, the data rate and the error rate increase. The data decoding errors are caused when the network jitter exceeds the assumed bound on the jitter. Note that whether the jitter is *i.i.d.* or not, does not affect the performance of the system as our *L-bit to n-packet* scheme is error-free for all bounded jitter. TCP can recover from packet loss, but retransmitting missing packets could result in decoding errors because of the jitter exceeding the assumed bound.

The setting in our experiments between Purdue and Priceton is comparable to that of [2]. As shown, when $\Delta = 40$ ms, $\delta = 10$ ms, the average decoding error rate between Purdue and Princeton is only 0.82%. The data rate of this timing channel is 42.75 bits/sec, which is more than twice the rate (16.76 bits/sec) in [2], while achieving higher accuracy (their error was 2%). When $\Delta = 10$ ms, $\delta = 10$ ms, the average decoding error between Purdue and Princeton is 4.06% and the standard deviation of the decoding error is 1.00%. In this case, we can

TABLE I
SUMMARY OF DECODING ERROR FOR THE TIMING CHANNEL EXPERIMENTS

Δ (ms)	δ ms	data rate (bits/sec)	Princeton mean(%)	stdev (%)	Stanford mean(%)	stdev (%)	Zurich mean(%)	stdev (%)	Madrid mean(%)	stdev (%)	Tsinghua mean(%)	stdev (%)
50	10	36.85	0.82	0.12	4.27	1.70	3.01	0.34	3.74	1.59	5.51	0.70
50	5	42.92	6.15	3.10	12.19	3.73	4.68	0.52	9.94	7.24	5.88	1.37
40	10	42.75	0.82	0.11	3.10	1.02	3.93	0.75	4.41	1.37	5.19	0.99
40	5	51.14	5.12	1.88	11.51	2.88	4.66	1.02	7.03	5.38	5.06	0.76
30	10	50.90	1.46	0.50	4.49	0.51	3.96	0.48	4.27	1.48	5.53	1.18
30	5	63.24	5.00	1.24	10.41	1.86	4.63	0.97	7.06	4.76	4.44	0.69
20	10	62.87	2.59	0.55	5.18	0.92	4.48	0.85	6.18	4.52	5.03	0.54
20	5	84.15	5.72	1.47	9.20	1.65	3.95	0.96	6.78	5.69	4.39	0.73
10	10	82.21	4.06	1.00	5.96	0.85	5.33	0.89	6.52	3.18	5.81	0.93
10	5	124.28	6.16	1.49	9.69	2.45	4.45	0.82	11.58	11.00	5.29	1.05
		Average RTT (ms)		39.96		67.17		135.65		155.09		272.81

achieve a rate of 82.21 bits/sec, which is five times the rate of [2].

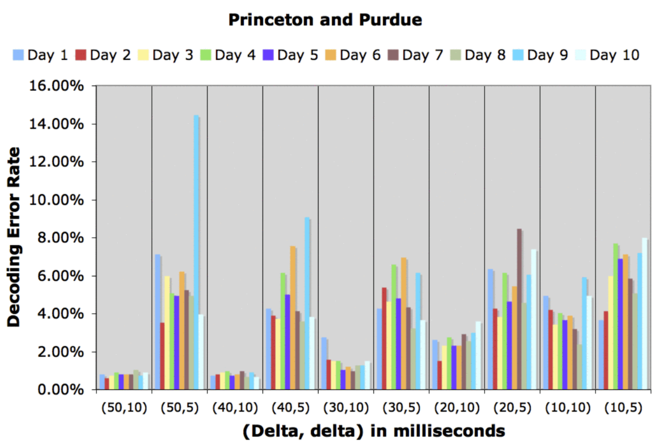


Fig. 6. Daily decoding errors for the covert channel between Purdue and Princeton

Figure 6 provides a detailed view of our daily experimental results between Princeton and Purdue. We notice there is an error spike (14%) on day 9 when $\Delta = 50$ ms, $\delta = 5$ ms. This could either be due to large variations in packet delays or packet losses on the network. We examined our log and compared the packet inter-transmission times T_i and inter-reception times R_i. We found the error is caused by the jitter of the network. In order to decode correctly, the combined jitter $|R_i - T_i|$ must be less than 2.5 ms for $\delta = 5ms$. In this experiment, 4.26% of packets have a combined jitter $|R_i - T_i| = 3ms$. and the jitter happens in random places. Since 3 packets map to 1 character, these 4.26% jitter results in 12.78% overall character decoding error. This also explains why the error rate is so small when $\delta = 10$, as it can tolerate combined jitter under 5 ms. The histogram of the combined jitter for day 9 is shown in Figure 8(a).

Our experiments with a receiver located outside the US also yielded good results. From Zurich and Purdue, all but one of the average decoding error are less than 5%. When $\Delta = 50$ ms and $\delta = 10$ ms, the average decoding error rate is only 3.01% and the data rate is nearly 37 bits/sec. Daily experimental results are illustrated in Figure 7.

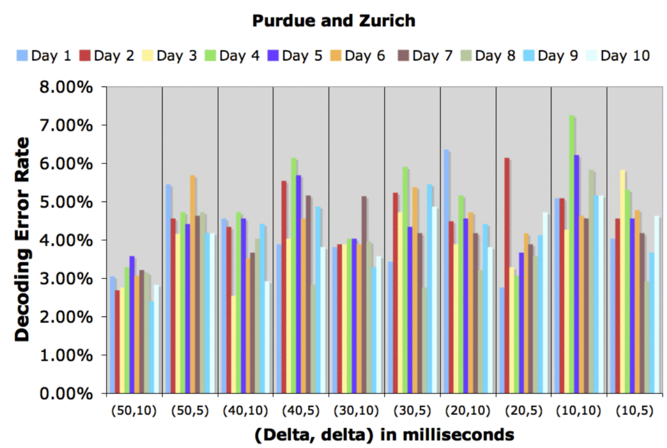
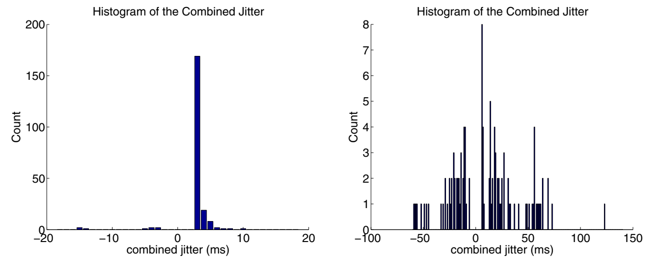


Fig. 7. Daily decoding errors for the covert channel between Purdue and Zurich

In addition to the 10 daily experiments we ran between the five pairs, we also ran the timing channel between Princeton and Purdue during various times of the day. We found that the network is more congested during the afternoon hours. The RTT can vary from 63 ms to 108 ms, and the combined jitter spreads more widely ranging from -50 ms to 120 ms. The decoding error thus increases to between 6% to 7%. The histogram of the combined jitter during this time is shown in Figure 8(b). In contrast to Figure 8(a) where the combined jitter is concentrated around 0, the combined jitter during busy hours spreads much more widely, and can range from -50 ms to 120 ms.

In these experiments, we demonstrated that our *L-bits to n-packets* scheme achieves good data rates with low error rate under various network conditions. However, this timing channel can also be easily detected, as the inter-transmission times T_i for our basic scheme are aligned on a grid of δ ms. One simple randomized scheme is for the sender and receiver to generate a common pseudo-random sequence $\{v_k, k = 1, 2, \cdots\}$ using a shared seed, v_k is uniformly distributed in



(a) Normal (Excluding $|R_i - T_i| < 3$) (b) Busy (Excluding $|R_i - T_i| < 5$)

Fig. 8. Combined Jitter during Normal and Busy Time

$(0, \delta)$. The sender adds the pseudo-random number v_k to the inter-transmission time T_k from the basic code. So, the inter-transmission time for this scheme is $T_k^{rand} = T_k + v_k$. Since the receiver knows the exact value of v_k for all k, it can decode just as in our basic scheme.

However, the above randomized timing channel can still be detected since the probability distribution of T_k^{rand} may not match any legitimate traffic pattern. In the next section, we will introduce a new scheme that allows the timing channel traffic to mimic a given legitimate traffic pattern. Moreover, we will show analytically that it is computationally impossible to detect our proposed covert channel usage using only the first order statistics of the packet inter-transmission time.

V. Non-detectable Timing Channel

The design goal of our non-detectable timing channel is for the timing channel traffic to be computationally indistinguishable from a class of legitimate traffic whose packet inter-transmission times are *i.i.d.*, while assuming bounded jitter. Telnet traffic is an example of such traffic, as its inter-transmission times can be modeled by an *i.i.d.* Pareto distribution [20].

A pseudo-random bit generator is called *secure* if an adversary cannot do better than random guessing at the next bit in the sequence from the prefix of the sequence. It has been proved that if a generator passes the next bit test, it will pass *all polynomial-time* statistical tests (Theorem 3.10 in [21]). Cryptographically secure pseudo random number generators (CSPRNG) such as Blum-Blum-Shub, Rabin, and RSA are provably secure PRNG. That is, they are able to generate pseudo random numbers that are computationally indistinguishable from true random numbers. On the contrary, *linear feedback shift registers*, a classical PRNG, is well known to be insecure. A thorough discussion on the theory of computational indistinguishability can be found in [21], [22].

Our non-detectable timing channel scheme is illustrated in Figure 9. In what follows, we will use an *8-bits to 2-packets* example to explain this scheme. In this example, an 8-bit ASCII character will be mapped into 2 packet inter-transmission times T_1, T_2 in three step. The shared code book contains the 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. These 256 vectors $(\frac{k_1}{16}, \frac{k_2}{16})$ are sufficient

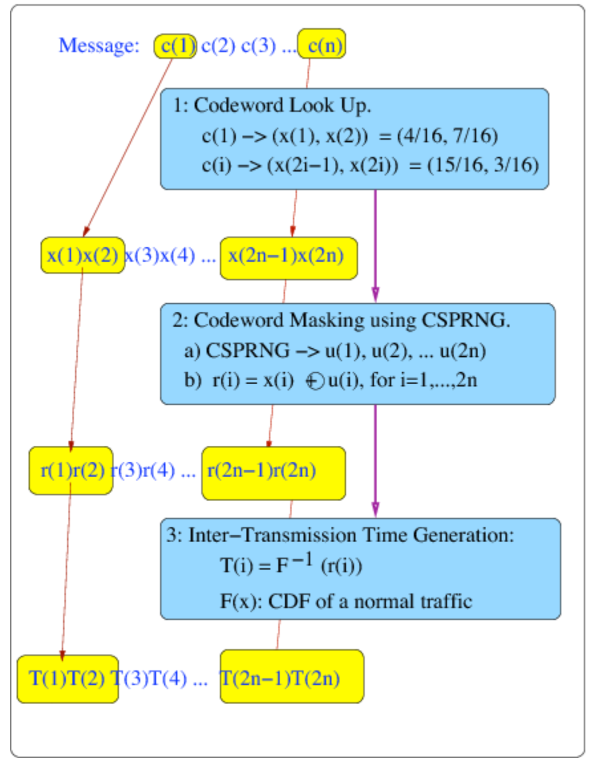


Fig. 9. Non-Detectable Scheme (Sender)

to accommodate all the 8-bit binary strings. Unlike our first scheme, the vector $(\frac{k_1}{16}, \frac{k_2}{16})$ does not directly correspond to any inter-transmission time, which will be obtained later.

Suppose the sender wishes to transmit a message consisting of a sequence of n characters $\mathbf{msg} = \{c_1, c_2, \cdots, c_n\}$. 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 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 $r_k = x_k \oplus u_k \triangleq x_k + u_k \bmod 1$. The *masking* can be thought of as the well-known *one-time pad* encryption technique operating on $\mathbf{x}$.

In the last step, we set $T_k = F^{-1}(r_k)$, where $F(x)$ is the given *c.d.f.* of the packet inter-transmission time of legitimate traffic. We use the sequence $T_1, T_2, \cdots, T_{2n}$ as the packet inter-transmission times for message $\{c_1, c_2, \cdots, c_n\}$. It can be shown that without knowing the seed, the sequence $T_1, T_2, \cdots, T_{2n}$ is computationally indistinguishable from a sequence of true *i.i.d.* random variables with *c.d.f.* $F(x)$, i.e., from legitimate traffic.

This computational indistinguishability can be proved using the framework of [21], [22]. The basic idea is the use of *proof by contradiction*. Suppose the following statement "**T** *is computationally indistinguishable from a sequence of true* i.i.d. *random variables with* c.d.f. $F(x)$." is **not true**. Then, we can find a polynomial time algorithm Q that can tell that the sequence $\mathbf{T}$ is not a sequence of true *i.i.d.* random variables with *c.d.f.* $F(x)$. Since $r_k = F(T_k)$ and $u_k = r_k - x_k \mod 1$, we have $u_k = F(T_k) - x_k \mod 1$. Then, we can construct another polynomial time algorithm Q* based on Q to tell that $u_1, u_2, \cdots, u_{2n}$ is not from a true *i.i.d.* uniform $(0, 1)$ random variables, which contradicts the construction that $u_1, u_2, \cdots$ is generated by a CSPRNG. Therefore, the statement *the sequence* $\mathbf{T}$ *is indeed computationally indistinguishable from a sequence of true* i.i.d. *random variables with* c.d.f. $F(x)$ must be true.

Another feature of our scheme is that it can mimic different traffic patterns using the same code book. Only the *c.d.f.* F(x) for the desired traffic pattern is needed in the last step to obtain $T_1, T_2, \cdots$ by setting $T_i = F^{-1}(r_i)$. This allows the sender and receiver to adapt to various traffic patterns easily. For instance, when the normal traffic pattern changes, the sender only needs to determine the *c.d.f.* of the distribution of the normal traffic, and it can use the new *c.d.f.* to map $\mathbf{r}$ to the desired packet inter-transmission times without changing the existing code book. Moreover, adaptation does not require any handshake between the sender and receiver, for the receiver can independently compute the *c.d.f.* using the traffic pattern of the inter-packet reception times.

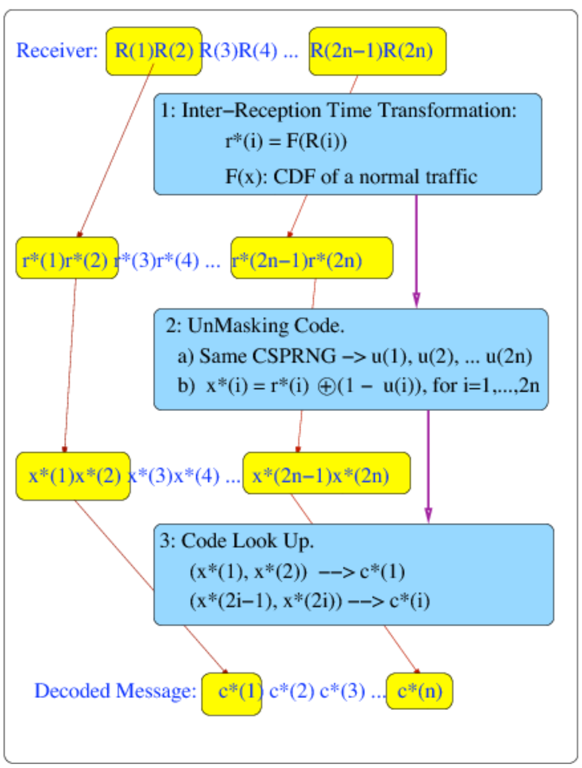


Fig. 10. Message Recovery (Receiver)

The procedure for recovering the message at the receiver is simply the reverse of the sender scheme, and is shown in Figure 10. Let $R_1, R_2, \cdots, R_{2n}$ be the packet inter-reception times. We first calculate $x_i^* = F(R_i) \oplus (1 - u_i)$ as depicted in step 1 and 2. In the last step, we first round x_i^* to the nearest value of $\frac{k}{16}$, denoted as x_i^d ($x_i^d = \lfloor 16 \cdot x_i^* + 0.5 \rfloor / 16$). We then decode (x_{2k}^d, x_{2k+1}^d) to character c_k^d by looking up the code book. The entire recovered message is then $c_1^d, c_2^d \cdots, c_n^d$.

The value of network jitter ϵ_i's and the $c.d.f.$ $F(x)$ of the legitimate traffic affect the decoding error, although the jitter does not need to be $i.i.d.$. Recall that $R_i = T_i + (\epsilon_i - \epsilon_{i-1})$, where $\epsilon_i - \epsilon_{i-1}$ signifies the combined jitter in the network and will be denoted as $\epsilon_i^{(c)}$. In this example, the receiver can decode correctly if $|\epsilon_i^{(c)}| < 1/(32 \cdot \sup |F'(x)|)$, where $F'(x)$ is the first order derivative of $F(x)$. This is because, $F(R_i) = F(T_i + \epsilon_i^{(c)}) = F(T_i) + F'(t^*) \cdot \epsilon_i^{(c)} = r_i + F'(t^*) \cdot \epsilon_i^{(c)}$, where $t^* \in (T_i, T_i + \epsilon_i^{(c)})$. Since $x_i = r_i \oplus (1 - u_i)$, we have $x_i^* = F(R_i) \oplus (1 - u_i) = x_i + F'(t^*) \cdot \epsilon_i^{(c)}$. Thus, $|x_i^* - x_i| < 1/32$, if $|\epsilon_i^{(c)}| < 1/(32 \cdot \sup |F'(x)|)$. This allows correct decoding (i.e. $x_i^d = x_i$), since x_i^d is the value of x_i^* rounded to the nearest $k/16$.

The example of *8-bit to 2-packet* scheme is readily generalized to an *L-bit to n-packet* scheme. The code book contains a one-to-one mapping of L-bit binary strings to n-dimensional vectors $(\frac{k_1}{K}, \cdots, \frac{k_n}{K})$, where K is a positive integer and $k_1, k_2, \cdots, k_n$ are non-negative integers smaller than K. Similar to the analysis for the above *8-bit to 2-packet* example, the value for K can be conservatively estimated as $K < ((\sup_x F'(x)) \cdot \epsilon_{max})^{-1}$ for correct decoding.

To demonstrate our non-detectable timing channel, we implemented an 8-bit to 2-packet scheme in which the timing channel traffic mimics the telnet traffic pattern. We use Java's *SecureRandom* class for the generation of cryptographically secure pseudo random numbers.

A Pareto distribution has a *c.d.f.*:

$$F(x) = P[X \leq x] = 1 - (\alpha/x)^{\beta}, \ x > \alpha, \alpha, \beta > 0$$

The inverse function of $F(x)$ needed in the step 3 is:

$$F^{-1}(x) = \alpha(\frac{1}{1-x})^{1/\beta}, \ 0 < x < 1$$

In our experiments, we use parameter $\alpha = 100ms$ and the shape parameter $\beta = 0.95$ as in [20]. The receiver is a PlanetLab node at Princeton University, and the sender is a host at Purdue University. We sent the same text file as in the previous experiments over this non-detectable timing channel. Our experiments were conducted during peak time when the RTT varies from 39.8 ms to 63.5 ms. The data rate is approximately 5 bits/sec, and the decoding error is only 1%. Figure 11 illustrates the distribution of the packet inter-transmission times from this experiment, along with Pareto and geometric distributions. We observe that visually the resulting traffic is similar to Pareto but distinct from geometrically distributed traffic. As the distribution of the inter-transmission time matches that of a given distribution, our scheme guarantees undetectability for any detector using only the first order statistics of the traffic. It is worth noting that the same concept can be applied to mimic traffic up to the second order (or even higher order) statistics. We are currently investigating efficient implementation that establishes the indistinguishability for higher order state-of-the-art detectors. For example, a recent covert channel detection scheme [16] uses entropy changes in correlated traffic to detect covert timing channels. Their detection method is based on the observation "that a covert timing channel cannot be created without causing some effects on the entropy of the original process". As the authors pointed out, this observation does not apply to *i.i.d.* processes as in our setting. An interesting future research direction is to derive a model for some normal traffic with correlated inter-packet transmission times, and to design a covert timing channel that mimics the correlated traffic.

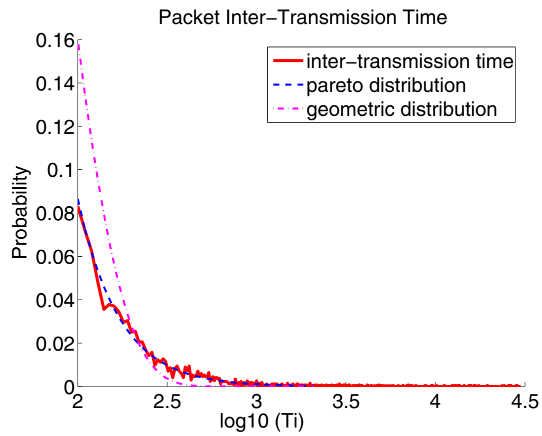


Fig. 11. Probability Distribution of T_i.

VI. Conclusion

We have designed a robust *L-bits to n-packets* scheme for communication using timing channels. The data rate of our scheme is close to the theoretical upper bound – the achievable rate of the geometric codes. We have implemented our scheme and have conducted extensive experiments on the PlanetLab nodes and found that our scheme achieves between two to five times the data rate of the previous state-of-the-art. In local networks with greater control over timing, one can significantly improve the achieved data rate. Thus, the data leakage rate can be much higher if the receiver is planted closer to the sender. We have also designed a computationally non-detectable timing channel scheme so that the distribution of the inter-transmission times generated by our timing channel mimics any *i.i.d.* traffic pattern. The non-detectable scheme results in a drop of the data rate; however it is still able to achieve a rate of 5 bits/sec for telnet traffic with only 1% decoding error even during peak time. This suggests that TCP/IP timing channels can be far stealthier than previously thought possible.

There are several interesting future directions for this work. One is to investigate the effect of jammers when additional jitter is added to the TCP/IP flow. Another is to design a computationally non-detectable covert timing channel that mimics correlated traffic.

VII. Acknowledgment

We thank Mr. Pablo Navarrete and Professor Edward Coyle for assisting us with our initial experiments. We thank Professor Ninghui Li for very helpful discussions on computational indistinguishability. This work has been supported in part by ARO No. W911NF-08-1-0238, NSF grants CNS-0523249 and CNS-0712846.

References

[1] U.S. Department of Defense. "Trusted computer system evaluation. The Orange Book," *DoD 5200.28-STD Washington: GPO:1985*, 1985
[2] 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
[3] V. Anantharam and S. Verdu, "Bits through queues," *IEEE Trans. Inform. Theory,* vol. 42, Jan. 1996
[4] J. A. Thomas "On the Shannon capacity of discrete time queues," *IEEE Int. Symp. Inform. Theory*, July 1997
[5] A. Bedekar and M. Azizoglu "The information-theoretic capacity of discrete-time queues," *IEEE Trans. Inform. Theory,* vol. 44, Mar. 1998
[6] J. Giles and B. Hajek "An Information-theoretic and game-theoretic study of timing channels," *IEEE Trans. on Inform. Theory,* VOL. 48, Sep. 2002
[7] X. Liu and R. Srikant "The timing capacity of single-server queues with multiple flows," *DIMACS Series in Discrete Mathematics and Theoretical Computer Science,* 2004
[8] A. B. Wagner and V. Anantharam "Information Theory of Covert Timing Channels," *Proceedings of the 2005 NATO/ASI Workshop on Network Security and Intrusion Detection*, 2005
[9] 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
[10] S. H. Sellke, C. C. Wang, N. B. Shroff, and S. Bagchi "Capacity bounds on timing channels with bounded service times," *in Proceedings of IEEE International Symposium on Information Theory*, June 2007
[11] I. S. Moskowitz and A. R. Miller "Simple timing channels," *Proceedings of IEEE Computer Society Symposim on Research in Security and Privacy*, 1994
[12] W. M. Hu "Reducing Timing Channels with Fuzzy Time," *in Proceedings of the IEEE Symposium in Security and Privacy*, May. 1991
[13] 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
[14] J. C. Wray "An analysis of covert timing channels," *Proceedings of IEEE Computer Society Symposim on Research in Security and Privacy*, May 1991
[15] V. Berk, A. Giani, and G. Cybenko "Detection of covert channel encoding in network packet delays," *Technical Report*
[16] S. Gianvecchio and H. Wang "Detecting Covert Timing Channels: An Entropy-Based Approach," *Proceedings of 14th ACM conf. Computer and communication security*, 2007
[17] G. Shah, A. Molina and M. Blaze "Keyboard and covert channels," *USENIX*, 2006
[18] H. Kopetz and G. Bauer "The time-triggered architecture," *Proceedings of the IEEE Special Issue on Modeling and Design of Embedded Software*, Jan 2003
[19] S. H. Sellke, C. C. Wang, S. Bagchi, and N. B. Shroff, "Covert TCP/IP Timing Channels: Theory to Implementation," Available: http://www.stat.purdue.edu/∼ssellke/CovertTC.pdf, 2007
[20] V. Paxson and S. Floyd "Wide-area traffic: the failure of Poisson modeling," *IEEE tran. Networking*, May 1991
[21] S. Goldwasser and M. Bellare "Lecture Notes on Cryptograpy," Available: *http://www-cse.ucsd.edu/ mihir/papers/gb.html*
[22] O. Goldreich *The Foundations of Cryptography*, *Cambridge University Press*, 2001