Cloak: A Ten-Fold Way for Reliable Covert Communications

Xiapu Luo, Edmond W.W. Chan, and Rocky K.C. Chang

Department of Computing
The Hong Kong Polytechnic University
{csxluo,cswwchan,csrchang}@comp.polyu.edu.hk

Abstract. In this paper, we propose *Cloak*—a new class of reliable timing channels—which is fundamentally different from other timing channels in several aspects. First, Cloak encodes a message by a unique distribution of N packets over X TCP flows. The combinatorial nature of the encoding methods increases the channel capacity largely with (N, X). Second, Cloak offers ten different encoding and decoding methods, each of which has a unique tradeoff among several important considerations, such as channel capacity and the need for packet marking. Third, the packet transmissions modulated by Cloak could be carefully crafted to mimic the normal TCP flows in a typical TCP-based application session. Although Cloak's basic idea is simple, we show in this paper how we tackle a number of challenging issues systematically. Our experiment results collected from PlanetLab nodes and a test bed suggest that Cloak is feasible under various network conditions and different round-trip delays.

Keywords: covert channel analysis, network security, attack models.

1 Introduction

In this paper, we consider data hiding techniques using network protocols as the cover. The communication channel under the cover is often referred to as a *network covert channel*. Network covert channels could pose a serious threat to the Internet security, because of their "proven" ability of stealthily exfiltrating stolen information (a hardware was built in [1]), coordinating an Internet-wide DDoS attacks [2] and Internet worm attack [3], coordinating a physical attack plan (a book was written about this possibility [4]), and other subversive operations. On the other *good* hand, they are useful for enhancing Internet privacy [5,6], watermarking encrypted flows in stepping stones [7], and tracking VoIP calls [8].

Similar to the classic covert channels in trusted computer systems, network covert channels could be classified into *storage channels* and *timing channels* [9,10]. In a storage channel, the encoder and decoder communicate covertly through "attributes of shared resources" [11], which could be any fields in a packet that can be "written" by the encoder and "read" by the decoder. The

J. Biskup and J. Lopez (Eds.): ESORICS 2007, LNCS 4734, pp. 283–298, 2007.

covert messages are encoded directly into these fields. Most existing network covert channels fall into this category. In a timing channel, the encoder and decoder communicate "through a temporal or ordering relationship of accesses to a shared resource" [11] which could be the timing of packet arrivals that can be modulated by the encoder and observed by the decoder. For example, an IP packet arrival within a time interval represents bit `1` and the absence of it represents bit `0` in an IP timing channel [12].

Existing network covert channels, however, suffer from low data rates in the presence of dynamic network conditions and *active network intermediaries* (ANI) (e.g., protocol scrubbers [13], traffic normalizer [14], and active wardens [15]). For example, the message encoding based on inter-packet delay is very sensitive to delay jitter, and packet losses affect the integrity of both timing and storage channels. On the other hand, storage channels do not suffer from these problems. Instead, their encoded messages could be altered by an ANI which modifies the replaceable header fields in the packets that pass through them.

In this paper, we propose *Cloak*—a new class of timing channels which is designed to be reliable under adverse network conditions. That is, Cloak's decoding accuracy is 100% even in the presence of packet losses, delay jitters, packet reordering, and packet duplications. The key elements responsible for this reliability property are using TCP data traffic as a cover (i.e., exploiting TCP's reliable transmission mechanism) and employing a fixed number of TCP packets (N) for encoding/decoding a message to avoid the inherent synchronization errors plaguing many network timing channels.

Another important deviation from other timing channels is that Cloak encodes a message with a unique distribution of N packets over X TCP flows, where $N, X > 1$. Due to the combinatorial nature of the encoding method, Cloak's channel capacity increases quickly with (N, X). Besides, Cloak offers ten different encoding and decoding methods. Each method tradeoffs among several conflicting design goals. Although Cloak uses multiple flows for the message encoding, the packet distribution over the flows can be carefully crafted to match with the normal TCP behavior in an application session. To our best knowledge, Cloak is the *first* network covert channel that exploits Enumerative Combinatorics [16] to convey hidden messages. Moreover, this original idea is generally enough for designing new covert channels and applying to other steganography problems.

The road map for the rest of this paper is as follows. Section 2 briefly discusses the previously proposed network timing channels. Section 3 presents the basic idea of message encoding in Cloak which is based on the well-known Twelvefold Way in the field of Enumerative Combinatorics. Section 4 details how we have resolved a number of difficult design issues for deploying Cloak in the Internet. Section 5 reports the test-bed and PlanetLab measurement results to evaluate Cloak's data rate under various network conditions and parameter settings. Section 6 summarizes this paper with a few venues of enhancing this work.

2 Related Work

Despite that information theorists have analyzed the capacity of covert timing channels for a long time, only recently have several practical timing channels emerged. On the network layer and above, there are so far two practical approaches to manipulating the packet timing: *ordered channels* and *inter-packet delay channels*. In the class of ordered channels, Kundur and Ahsan [17] propose to re-sort the original order of a flow of IPSec packets and use the out-of-orderliness to imbed messages. Chakinala et al. [18] further extend the approach to TCP packets and formalize various models for these ordered channels.

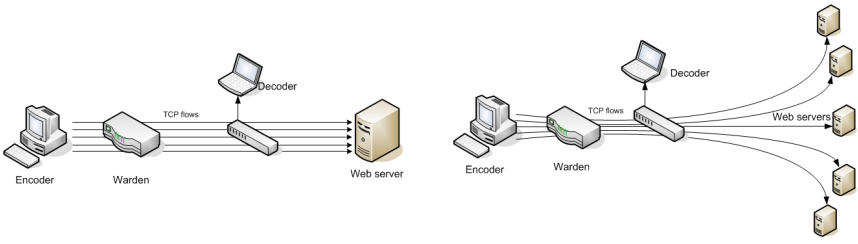
The class of inter-packet delay channels, on the other hand, embeds messages in the delay period between selected packets. Cabuk et al. [12] propose an IP timing channel, where an IP packet arrival during a timing interval is decoded as `1` and the absence of it is decoded as `0`. Shah et al. recently [1] propose JitterBug, another timing channel to encode binary bits. Unlike the IP timing channel, JitterBug encodes binary bits into the packet inter-arrival times, and it does not need to inject new packets. Moreover, they have presented a convincing threat of leaking keyboard typed secrets, such as passwords, through the timing channel and have built a hardware to demonstrate its feasibility. Berk et al. [19] have considered using inter-packet delay of ICMP packets to encode one or multiple bits. For example, bit `1` is encoded by a longer inter-packet delay, whereas bit `0` is encoded by a smaller inter-packet delay.

3 The Basic Idea

3.1 Encoding Based on Packet-Flow Distributions

The covert messages in Cloak are encoded by a class of combinatorial objects—each covert message is encoded with a unique distribution of N TCP packets over X TCP flows. The encoder and decoder agree on the values of N and X beforehand. Furthermore, the encoder will transmit the next message *only* after receiving the ACKs for the message just sent. On the other side of the channel, the decoder starts decoding as soon as collecting N TCP packets from the encoder. Moreover, the encoder and decoder do not have to explicitly exchange the "codebook"; as will show in section 4.1, the encoding and decoding can be performed using unranking and ranking functions.

It is worthwhile to point out here that Cloak is reliable in the same sense of reliability in TCP even when the messages experience adverse network conditions. First of all, Cloak's decoding accuracy is not affected by delay jitters, because the encoding is not based on the actual time. Second, since the encoder sends a covert message one at a time, it can detect whether the decoder has successfully received the last message based on the ACKs for the N TCP packets. Upon detecting an unsuccessful reception, the encoder could "partially" resend the message. The decoder, on the other hand, will decode only after receiving N in-sequenced TCP packets from the encoder. Therefore, if Cloak is implemented using the normal TCP stack, no additional reliability mechanism is needed to guarantee Cloak's reliability.



(a) The five TCP flows connect to the same Web server.

(b) The five TCP flows connect to different Web servers.

Fig. 1. Two covert communication scenarios between Cloak encoder and decoder

In Figure 1, we depict two different scenarios for the Cloak encoder and decoder to communicate. In both cases, we assume a warden on the encoder's network who guards against any network covert channels initiated from inside. The warden could be active or passive. In the first scenario (Figure 1(a)), the encoder establishes a "normal" HTTP session with a remote server which consists of five TCP flows. The encoder encodes the messages into the TCP flows; the decoder eavesdrops at any point of the path and decodes the messages. Moreover, the warden could not detect Cloak simply based on the presence of multiple TCP flows to the same server, because it is not uncommon to have multiple TCP flows in an HTTP session. Moreover, multi-thread upload or download (i.e., sending commands) also has similar traffic patterns.

In the second scenario (Figure 1(b)), the encoder establishes normal HTTP sessions with multiple servers which are dispersed at different locations. Therefore, the decoder should be located on the common routing path for all the servers. Although this approach restricts the decoder location, it can diffuse the relationship among the TCP flows. A simple approach for relaxing this restriction is to use distributed information collection, for example, through a botnet. Each bot will observe partial information and then sends it to the commander.

3.2 The Twelvefold Way

Besides the encoding algorithm just described, Cloak could admit other encoding methods. In fact, Cloak offers ten different encoding methods which are based on the well-known *Twelvefold Way* [16] in the field of Enumerative Combinatorics. The Twelvefold Way refers to twelve basic counting problems that count all the possible ways of putting N balls into X urns, and their results. Let the set of balls be $\mathbb{N}$ ($|\mathbb{N}| = N$) and the set of urns be $\mathbb{X}$ ($|\mathbb{X}| = X$). Each problem can be based on whether the balls and urns are distinguishable or not (e.g., by their colors), and three possible kinds of ball distributions over the urns: (1) no restriction, (2) at most one ball per urn, and (3) at least one ball per urn. These three cases can be equivalently represented by an arbitrary function $\mathbf{f}_A : \mathbb{N} \to \mathbb{X}$, an injective function $\mathbf{f}_I : \mathbb{N} \to \mathbb{X}$, and a surjective function $\mathbf{f}_S : \mathbb{N} \to \mathbb{X}$, respectively.

The correspondence between balls and urns, and packets and flows is obvious. Table 1 summarizes the Twelvefold Way using flows (urns) and packets (balls)

[16]. Each of the twelve results answers the corresponding counting problem (i.e., the total number of unique packet-flow distributions). Cases (11) and (12) obviously cannot be used in Cloak, therefore the ten encoding methods. In the rest of this paper, we refer the ten cases to as Cloak$^c(N, X)$, $c \in [1, 10]$. Due to the space limitation, we refer to [20] for the proofs of the results in Table 1.

Table 1. The Twelvefold Way and their relation to the ten (items 1-10) encoding methods in Cloak

Elements of $\mathbb{N}$ (TCP packets)	Elements of $\mathbb{X}$ (TCP flows)	$\mathbf{f}_A$ (no restriction)	$\mathbf{f}_I$ (at most 1 packet in a flow)	$\mathbf{f}_S$ (at least 1 packet in a flow)
Distinguishable	Distinguishable	X^N (1)	$N!C_X^N$ (2)	$X!S(N, X)$ (3)
Indistinguishable	Distinguishable	C_{N+X-1}^{X-1} (4)	C_X^N (5)	C_{N-1}^{X-1} (6)
Distinguishable	Indistinguishable	$\sum_{i=1}^{X} S(N, i)$ (7)	$\begin{cases} 1 \text{ if } N \leq X \\ 0 \text{ if } N > X \end{cases}$ (11)	$S(N, X)$ (8)
Indistinguishable	Indistinguishable	$\sum_{i=1}^{X} P(N, i)$ (9)	$\begin{cases} 1 \text{ if } N \leq X \\ 0 \text{ if } N > X \end{cases}$ (12)	$P(N, X)$ (10)

where
– $C_X^N = \frac{X!}{N!(X-N)!}$ and $S(N, X) = \frac{1}{X!}\sum_{j=1}^{X}(-1)^{X-j}C_X^j j^N$.
– $P(N, X)$ is the number of partitions of N into X parts.

According to Table 1, some encoding methods require distinguishable packets and/or distinguishable flows. The correspondence between the ball and urn distinguishability, and the flow and packet distinguishability is somewhat tricky. First of all, all TCP flows and packets are of course distinguishable. However, the original counting problems assume that the colors of the urns and balls do not change, but this is not the case for Cloak. For instance, the "marking information" in the flows and packets could be altered by an ANI. Therefore, the TCP flows (or packets) are considered distinguishable only if both encoder and decoder are able to identify the same flow (or packet).

3.3 The Ten-Fold Way in Cloak

In this section, we discuss the differences among the ten encoding methods and explain why we need all of them. The first important difference among them is their channel capacity. By modeling a Cloak channel as a classical information channel, we can obtain the capacity of a Cloak$^c(N, X)$ channel in bits/symbol based on the mutual information [21]. Since Cloak is reliable and there is only one set of covert messages, the channel capacity can be increased only by increasing the size of the covert message set. By denoting the Twelvefold Way result for Cloak$^c(N, X)$ by $T^c(N, X)$, a higher value of $T^c(N, X)$ therefore gives a higher channel capacity. Furthermore, each unique packet-flow distribution can encode an L-bit word, where $1 \leq L \leq \lfloor \log_2 T^c(N, X) \rfloor$.

In the following, we explain the relationships between the channel capacity and the flow and packet distinguishability. First, making the flows distinguishable increases the channel capacity (e.g., $T^1(N, X) > T^7(N, X)$). Similarly, making

the packets distinguishable also increases the channel capacity (e.g., $T^1(N,X) > T^4(N,X)$). Finally, for each row in Table 1, the channel capacity for $\mathbf{f}_A$ is the largest, e.g., $T^1(N,X) > T^3(N,X)$, and $T^7(N,X) > T^8(N,X)$. Based on the channel capacity, we define *data rate* in bits/second as $\frac{C}{T_s}$, where T_s is the time for transmitting a message. The minimal time for transmitting a message in Cloak (i.e., the N packets in X flows) is one round-trip time (RTT) between the encoder and decoder. To achieve a reasonable channel capacity, we therefore consider $X > 1$ and $N > 1$ in the rest of this paper.

Besides the channel capacity, the ten encoding methods differ also in three other important aspects. The first one concerns the channels that require distinguishable packets (i.e., $c = 1, 3, 7, 8$). For these channels, the encoder usually adds "markers" to the TCP packets in order to make them distinguishable. The additional markers, however, could be "modified" when the packets traverse an active warden, which could result in decoding errors. In other words, there is a tradeoff between achieving a higher channel capacity by making the packets distinguishable and the decoding accuracy. Similar problems occur also to the channels with flow distinguishability. We have discussed how to make packet or flow distinguishable in the full paper [20].

The second one is connected to a head-of-line blocking (HoLB) problem. To explain the issue, consider $c = 1, 2$. the difference between them is that the second method caps the number of packets distributed to a flow to one. Therefore, in terms of the packet distribution, the flows for $c = 2$ differ at most by one packet, but that for $c = 1$ is N (i.e., all the packets are distributed to a single flow). The latter case may require several RTTs to complete the transmission of a message; thus, this HoLB problem, as we shall see later, could reduce the actual data rate significantly. The last issue is that some flows for the methods under $\mathbf{f}_A$ and $\mathbf{f}_I$ may become idle for a prolonged period of time, which may cause the remote servers to close the connection. However, those methods under $\mathbf{f_S}$ mitigate this problem by insisting each flow to carry at least one packet for each message.

4 Design Issues

In this section, we discuss a number of design elements that are central to a practical deployment of Cloak in the Internet and to Cloak's performance.

4.1 Message Encoding and Decoding

As mentioned in the last section, the encoder and decoder do not need to exchange a codebook explicitly. Instead, they use two special functions for encoding and decoding: `Rank()` and `Unrank()`. Each Cloak$^c(N,X)$ channel has its own function pair. The function `Rank()` takes in a flow-packet distribution and returns its *rank* that is the index of the flow-packet distribution in the decreasing lexicographically ordered array of all possible distributions, staring from 0. `Unrank()` does the opposite—taking in a rank and returning the corresponding flow-packet distribution.

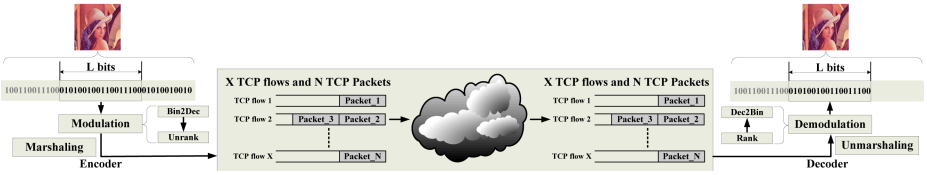


Fig. 2. The encoding and decoding processes in Cloak

Figure 2 depicts the encoding and decoding processes in Cloak. The encoder and decoder are assumed to have agreed on (c, N, X) beforehand. They could also dynamically change (c, N, X) by exploiting the random beacons widely available in the Internet. The messages are encoded based on L-bit words, where $1 \leq L \leq \lfloor \log_2 T^c(N, X) \rfloor$. There are three major steps involved in sending a covert message. Each L-bit word is first converted to the nonnegative decimal value (through the `Bin2Dec()` function) that serves as the rank for the corresponding packet-flow distribution. Then, `Unrank()` is invoked to compute the distribution. Finally, the encoder marshals the packet-flow code into the actual TCP flows and data packets. After sending the N packets over the X flows, the encoder has to receive the ACKs for the N packets before sending the next N packets. In the case of packet losses, Cloak may rely on TCP to recover them.

The three-step process above is exactly reversed for receiving a covert message. In the first step, the decoder unmarshalls the packet-flow distribution from the flows and packets received from the encoder. That is, the decoder collects exactly N TCP packets from the X flows before moving to the next step. Moreover, since the number of flows can be distinguished based on the order of the TCP three-way handshaking performed, the decoder can count the number of data packets in each flow. Similar as before, any TCP packet loss, duplication, or reordering can be taken care of by TCP. As soon as N packets are collected, the decoder feeds the distribution into `Rank()` which yields the corresponding rank. As a last step, the rank is converted back to the L-bit word (through the function `Dec2Bin()`). We refer the detailed ranking and unranking algorithms to the full paper [20].

4.2 A Head-of-Line Blocking Problem

In this section, we discuss a head-of-line blocking (HoLB) problem that we have encountered when conducting Internet experiments. The HoLB problem degrades the data rates of all encoding methods, except for $c = 2, 5$. To explain the problem, we consider an extreme scenario where most of the N packets are distributed to a single flow, while other flows receive at most one packet. Therefore, the total transmission time for the message is governed by the time required to transmit the packets in the most busy flow which prevents the encoder from transmitting the next message. Furthermore, since the TCP congestion window usually starts with one or two packets, it will take the busy flow's sender several RTTs to complete the transmissions, thus leading to a low data rate. The

problem may become worse if there are packet losses in the most busy flow that will retransmit those packets according to the timeout mechanism or the fast retransmission/fast recovery mechanism. This issue will also occur to the flows that are connected to different servers which experience a wide range of RTTs.

A simple way of mitigating the HoLB problem is to aggressively transmit every N packets. The basic idea is that the encoder will dispatch all packets belonging to kth message after receiving ACK packets that acknowledge the data packets for the $(k-1)$th message or a timer with period T_E expires. If the encoder does not receive all the expected ACK packets before T_E, it will retransmit unacknowledged packets and reset the timer. T_E is usually set to the estimated RTT that is computed through the exponential weighted moving average (EWMA) of RTT samples, an approach similar to the one used in normal TCP. However, the downside is that the resulting traffic pattern will be different from normal TCP behavior. This has prompted us to design a new codeword scheme to be discussed next.

A *D*-limited codeword scheme. The D-limited codeword scheme essentially caps the maximum number of packets assigned to a flow to D; that is, it enforces $\max\{n_i\} \leq D$, where $D \geq 1$ is a constant. The choice of D should be chosen such that it is less than the encoder's TCP send window size in terms of packets. In this way, all the packets can be sent out in one RTT; otherwise, multiple RTTs would be needed for transmitting a message.

We use $c = 10$ (indistinguishable packets and flows) to illustrate how this codeword scheme works. We first define the following two quantities:

1. Let $\Upsilon(N)$ be the total number of ways to distribute N packets into TCP flows such that each flow is given *at most* D packets.
2. Let $\Gamma(N, D)$ be the total number of ways to distribute N packets into D flows such that each flow is assigned at least one packet. Note that $\Gamma(N, D) = P(N, D)$ if both packets and flows are indistinguishable (i.e., $c = 10$).

Theorem 1. *If both packets and flows are indistinguishable,* $\Upsilon(N) = \sum_{i=1}^{D} P(N, i)$.

Corollary 1. *To generate D-limited codewords from $P(N, D)$, we need at most $N + 1 - D$ flows to convey a message.*

Theorem 1 computes how much information this D-limited codeword scheme could transmit. Corollary 1, on the other hand, shows that if the upper bound on the number of flows is X, then $N \leq X + D - 1$. Their proofs are given in [20]. We now use Proposition 1 and Corollary 1 directly to construct D-limited codewords for $c = 10$:

1. **Encoding.** To transmit a message (a binary string), the encoder first calculates its decimal value and then uses Cloak[10]'s unranking algorithm to get the corresponding packet-flow distribution, denoted by ζ. After that, the encoder computes ζ's conjugate [16], denoted by ζ', and transmits packets according to ζ'.

2. **Decoding.** Upon receiving a packet-flow distribution ζ', the decoder first computes its conjugate ζ and then uses Cloak$^{10}(N, X)$'s ranking algorithm to decode the message.

To construct D-limited codewords for other encoding methods, we could adopt our general framework for designing new ranking and unranking algorithms [20]. That is, when the encoder receives ζ' from Cloak^9 or Cloak^{10}, it could expand ζ' by considering distinguishable packets or flows. For example, if only flows are distinguishable, we could permute the locations of flows that have different n_i and then increase the capacity in a way similar to $\lambda!$ or C_X^N. If only packets are distinguishable, we could consider how to partition them into different flows and therefore to increase the capacity in a way similar to $S(N, X)!$. If both flows and packets are distinguishable, we could permute the locations of packets that belong to different flows. The only requirement is not to change the value of n_i.

5 Experimental Results

In this section we discuss how Cloak's data rate is affected by the RTT, router hop distance, geographical locations, and various adverse network conditions. Besides, we evaluate the effect of the HoLB problem on Cloak, and its performance with the D-limited codeword scheme. We also compare Cloak's performance with other timing channels: IP timing channel (IPTime) [12] and JitterBug [1], wherever we find appropriate. We have conducted experiments in the real Internet environment using the PlanetLab platform, and our test-bed which permits controlled experiments configured with various network conditions. Here we present experiment results obtained from the Planetlab platform and leave the results from the test bed to [20].

We measure the data rate of the timing channels in terms of their *goodput* defined as:

$$G = (1 - p_e)\frac{M \times L}{T_d}, \tag{1}$$

where T_d is the total time required for delivering M L-bit covert messages, and p_e is the channel's bit error rate (BER). The BER is computed based on the *Levenshtein distance* which is given by the number of insertions, deletions, and substitutions needed to convert a source message into a decoded message. Since Cloak is reliable, its p_e is 0.

5.1 Implementation

We have implemented Cloak's encoder and decoder as a TCP client and a TCP listener, respectively, including the ten `Rank()` and `Unrank()` functions. We have implemented `Bin2Dec()`, `Dec2Bin()`, `Rank()`, and `Unrank()` as offline functions. That is, the encoder pre-computes all the packet-flow combinations, and the decoder starts decoding only after capturing all N packets from X flows.

Cloak. For the Cloak's encoder, we have implemented two types of transmission functions based on the TCP socket (Cloak(STREAM)) and the raw socket (Cloak(RAW)). In Cloak(STREAM), the system's TCP stack guarantees the transmission reliability, and its traffic pattern resembles normal TCP flows'. However, it may take several RTTs to complete a single codeword transmission, thus limiting its data rate. Cloak(RAW), on the other hand, applies the aggressive transmission mechanism discussed in section 4.2 to improve its data rate. We have also implemented a separate capturing thread in the encoder to monitor the ACK arrivals, in order to determine if the other side has received all the N packets. We have implemented the Cloak's decoder with `libpcap` v0.9.5 library to sniff TCP packets. Moreover, we use a snaplen of 96 bytes to reduce the overhead during the packet capturing operation. We did not observe any packet drops throughout the experiments.

IPTime and JitterBug. We have implemented both IPTime's and JitterBug's encoding and decoding schemes as plug-in modules in the Cloak encoder and decoder, respectively. We employ UDP socket (i.e., `SOCK_DGRAM`), because the packet transmission in these two timing channels do not require reliability. During the encoding process, the plug-ins invoke the modulation function in the Cloak encoder to let the codeword bypass `Bin2Dec()` and `Unrank()`, and to marshal the binary stream directly into a flow of modulated UDP packets. Moreover, the encoder generates the modulated sequences complying with the specifications of IPTime and JitterBug. Both the IPTime's encoder and JitterBug's encoder use a fixed timing interval (or timing window) of w. The JitterBug's encoder, in addition, has a default tolerance parameter of $\varepsilon = w/4$. The corresponding plug-ins in the decoder perform the reverse procedures for decoding. Moreover, we did not implement any framing and error correction mechanism for Cloak, IPTime, and JitterBug.

5.2 The Setup of PlanetLab Experiment Platforms

We locate the encoders in nine geographically diverse PlanetLab nodes, and the decoders and a Web server in a campus network. The encoders send packets to the Web server, and the decoder eavesdrops the packets and decodes them. We have obtained a total of 17,545 RTT samples between the decoder and each PlanetLab node during the experiment period. Table 2 shows the nine PlanetLab nodes with the router hop counts from the encoder to them and the RTT statistics with a 95% confidence interval. Note that the average RTTs range between 0.0652 seconds and 0.3418 seconds. Moreover, the RTT measurements for JP, KR, and CA have higher variations than the others.

5.3 PlanetLab Experiments

Experiment design. To observe the page limitation, we report experiment results only for Cloak$^1(N, X)$. To study the effect of N, we fix X to 20 to give a large enough number of flows, and $N = \{5, 9, 10, 11, 15, 20, 30, 40, 50\}$ which covers a reasonable range of channel capacity. Similarly, to study the effect of X, we fix N to 20 and consider $X = \{4, 6, 8, 10, 12, 14\}$.

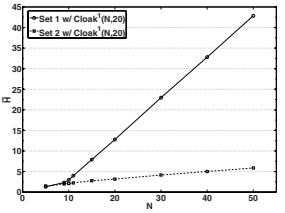
Table 2. Measured path characteristics between each PlanetLab site and the decoder machine

Locations	Hops	RTT		
		Means	Std. Dev.	95% Conf. Intervals
Shenyang, China (CN)	13	.0652	.0060	.0651/.0653
Tokyo, Japan (JP)	16	.0992	.0244	.0988/.0996
California, U.S. (CA)	14	.1767	.0230	.1763/.1770
Kansas, U.S. (KS)	16	.2176	.0056	.2175/.2177
Rhode Island, U.S. (RI)	13	.2267	.0074	.2266/.2268
Gwangju, Korea (KR)	18	.2343	.0356	.2338/.2348
Ghent, Belgium (BE)	16	.3075	.0048	.3074/.3075
London, UK (UK)	19	.3124	.0061	.3123/.3124
Lisbon, Portugal (PT)	17	.3418	.0171	.3415/.3420

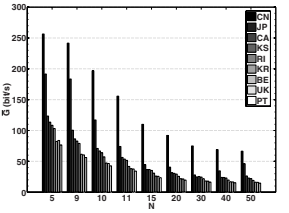
To study the adverse effects of the HoLB problem, we have generated two sets of codewords (datasets 1 and 2) for each N in Cloak$^1(N, 20)$. Each dataset consists of 100 L-bit ($M = 100$ and $L = \lfloor \log_2 X^N \rfloor$) codewords. Moreover, we assign each packet in dataset 2 to the 20 flows with equal probability; however, we intentionally assign more packets in dataset 1 to flow 1. We measure the *degree of HoLB* of a codeword by $H = \max_{0 \le i < X} n_i$. Figure 3(a) plots the values of $\overline{H}$, the mean values of H for different values of N. As shown, the rate of increase in $\overline{H}$ for dataset 1 is about 10 times higher than that for dataset 2 when N is beyond 10. Moreover, we have generated other sets of codewords (datasets 3 and 4) for each X in Cloak$^1(20, X)$. The codewords for datasets 3 and 4 are generated the same ways as for datasets 1 and 2, respectively. Figure 4(a) shows that the values of $\overline{H}$ for the two datasets diverge as X increases.

Experiment results. Figures 3(b), 3(c), 4(b), and 4(c) plot the average goodputs for the nine PlanetLab nodes with the four datasets of codewords. We compute the average goodput for each (N,X) tuple by performing 30 measurements. For each N or X, the nine nodes in the figures are sorted in the ascending order of their measured mean RTTs given in Table 2. We first report the results for datasets 2 and 4 (Figure 3(c) and Figure 4(c)) for which the packets are assigned uniformly to the 20 flows. Among all the nodes, CN achieves a maximum channel goodput of around 450 bit/s in Figure 3(c). Both figures also show that the average goodput $\overline{G}$ for the two smallest RTTs (nodes CN and JP) are the highest. However, the goodputs do not necessarily decrease with the RTTs. That is, although the goodputs are inversely proportional to the RTTs, there are other factors, such as packet losses, that could disturb the goodputs. Moreover, the increase seems to be more drastic for the case of increasing X. For example, the JP node's goodput is increased by more than four times as X increases from 4 to 12. On the other hand, the rates of increases for other nodes with longer RTTs are smaller. That is, a large RTT will reduce the gain obtained from the increase in the channel capacity.

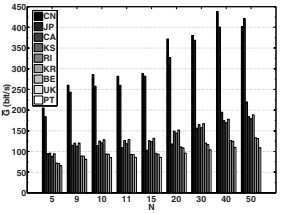
Next, we evaluate the effects of the biased packet distributions on the average goodput. We first compare the results for datasets 1 and 2 (Figure 3(b)



(a) The values of $\overline{H}$ for datasets 1 and 2 as a function of N.

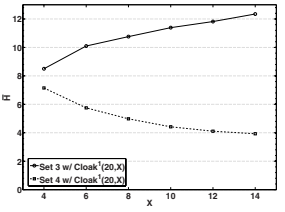


(b) The average goodput for the PlanetLab nodes with dataset 1.

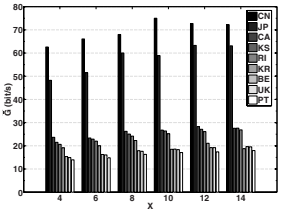


(c) The average goodput for the PlanetLab nodes with dataset 2.

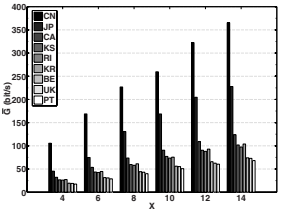
Fig. 3. The results for the PlanetLab nodes: the average goodput verses N for Cloak$^1(N, 20)$ with datasets 1 and 2



(a) The values of $\overline{H}$ for datasets 3 and 4 as a function of X.



(b) The average goodput for the PlanetLab nodes with dataset 3.

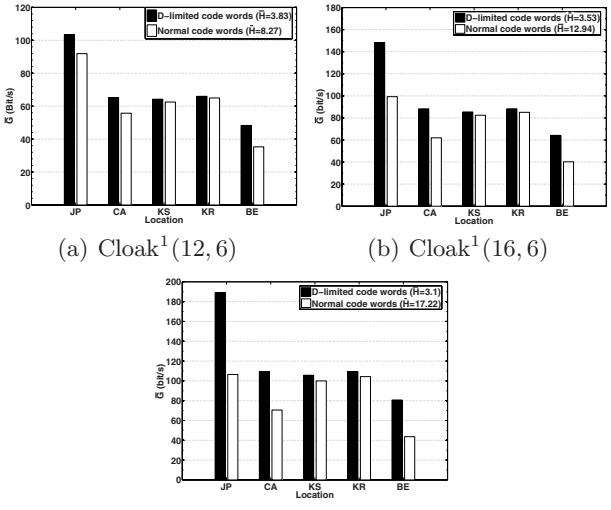


(c) The average goodput for the PlanetLab nodes with dataset 4.

Fig. 4. The results for the PlanetLab nodes: the average goodput verses X for Cloak$^1(20, X)$ with datasets 3 and 4

and Figure 3(c)). The comparison reveals that they show opposite trends as N increases: the goodput decreases with N in Figure 3(b). It is important to point out that the scales of the two figures are actually different. Therefore, the goodputs in Figure 3(c) are all greater than the respective cases in Figure 3(b), except for $N = 5$. Since $\overline{H}$ increases with N as shown in Figure 3(a), it will take flow 1 a longer time to complete its packet transmission as N increases. For the comparison of datasets 3 and 4 (Figure 4(b) and Figure 4(c)), the goodputs in Figure 4(c) are all greater than the respective cases in Figure 4(b). However, unlike the previous cases, the goodputs in Figure 4(b) slightly improve as X increases, but the goodputs stop growing as X reaches 10. An increase in X in fact alleviates the HoLB problem, because flow 1 will become less busy; as a result, it is not surprising to see some improvement in the goodputs as X increases.

Evaluation of the D-limited codewords. To measure the performance of the D-limited codewords, we have selected five (JP, CA, KS, KR, and BE) out of the nine PlanetLab nodes to measure the average goodput of Cloak. Similar to the last section, we have generated a set of 100 L-bit binary codewords for each (N, X) tuple for Cloak$^1(N, X)$, where $X = 6$ and $N = \{12, 16, 20\}$. We



(a) Cloak$^1(12, 6)$ (b) Cloak$^1(16, 6)$

(c) Cloak$^1(20, 6)$

Fig. 5. Comparing the average goodput for the normal codewords and the 6-limited codewords

use Cloak(STREAM) to encode them into two distinct sets of codewords: one generated by the D-limited codewords scheme with $D = 6$ and the other by the normal codewords. The average goodput is again based on 30 measurements.

Figure 5 compares the average goodputs of the two codewords for the five nodes. The figures show that the D-limited codeword always gives a higher goodput than the normal scheme for all nodes and for all three (N, X) tuples. Each figure also gives the average degrees of HoLB for the two codewords. The average degrees for the D-limited codewords are quite stable in all three cases, whereas the degree for the normal codewords is the highest in Figure 5(c), followed by Figure 5(b) and then by Figure 5(a). As a result, the percent of improvement of using the D-limited codewords also follows the same decreasing order for nodes JP, CA, and BE in Figures 5(a)-5(c). In particular, we have noticed a maximum gain of 77% from the JP node with Cloak$^1(20, 6)$. On the other hand, the nodes KS and KR attain much less gains; for example, the gain is only 1.6% for the KR node with Cloak$^1(12, 6)$. By examining the traffic traces, we have found that the packet loss rates at these two nodes are much lower than the others. Therefore, the normal scheme has already achieved a very high goodput; the additional benefit of adopting the D-limited scheme becomes marginal. We have also evaluated the performance of the aggressive transmission scheme and found that it could significantly increase Cloak's goodput [20].

Comparing Cloak, JitterBug, and IPTime. We have also conducted experiments on JitterBug and IPTime in the five PlanetLab nodes. In this set of experiments, we have generated another 100 packet-flow codewords using the normal Cloak$^1(20, 4)$ encoder with $\overline{H} = 5.86$. Each node uses both Cloak(RAW)

Table 3. The average goodput and average BER for Cloak, IPTime, and JitterBug obtained from five PlanetLab nodes

	95% confidence intervals of average goodput (average BER)		
	Cloak(`RAW`)	Cloak(`STREAM`)	JitterBug(RTT)
JP	203.86/216.28 (0)	55.17/58.57 (0)	13.01/13.04 (.0155)
CA	85.66/88.49 (0)	68.75/71.75 (0)	7.33/7.35 (.0265)
KS	90.77/91.00 (0)	66.57/69.26 (0)	6.13/6.14 (.0018)
KR	106.52/107.90 (0)	68.68/71.51 (0)	5.67/5.68 (.0012)
BE	63.88/64.20 (0)	44.69/45.61 (0)	4.36/4.36 (.0011)
	JitterBug(1.5RTT)	IPTime(RTT)	IPTime(1.5RTT)
JP	8.77/8.78 (.0164)	9.48/9.50 (.0363)	6.49/6.51 (.0209)
CA	4.76/4.81 (.0521)	5.43/5.47 (.0282)	3.68/3.69 (.0158)
KS	4.10/4.10 (.0010)	4.50/4.51 (.0112)	3.02/3.02 (.0088)
KR	3.81/3.81 (.0010)	4.17/4.18 (.0126)	2.80/2.81 (.0081)
BE	2.91/2.91 (.0007)	3.21/3.21 (.0076)	2.14/2.15 (.0066)

and Cloak(STREAM) to transmit the codewords. We set Cloak(RAW)'s T_E to the measured mean RTTs. For the JitterBug and IPTime experiments, the encoder marshals each respective binary codeword directly into a flow of modulated UDP packets with $w = \{\text{RTT}, 1.5\text{RTT}\}$. Both the average goodput and average BER are computed based on 30 samples.

We summarize the experiment results in Table 3. In each cell, the two leftmost values correspond to the lower limit of and the upper limit of the 95% confidence intervals for the same average goodput, and the rightmost value inside the parentheses corresponds to the measured average BER. We first point out that it is difficult to conduct a fair comparison among the three channels, because, for example, Cloak uses multiple flows whereas the other two use only one. Therefore, the comparison is based on how their goodputs are affected by the RTTs. Recall that the five nodes are sorted in an ascending order of their mean RTTs. For both Cloak channels, we do not find any general relationship between their goodputs and the RTTs, except that the lowest goodputs for both cases are given by the highest RTT (i.e., BE). On the other hand, the goodputs for JitterBug and IPTime show downward trends as the RTT increases. The magnitude of the goodput degradation is rather significant, which is between three to four times when comparing the goodputs for JP and BE. Their average BERs also show similar downward trends except for a couple points.

6 Conclusions and Future Work

In this paper, we propose Cloak, a new class of timing channels. The major design choices responsible for Cloak's attractive properties are the use of TCP

as the cloaking medium, and the exploitation of Enumerative Combinatorics to encode a message into multiple TCP flows and a fixed number of TCP packets. The former provides the needed reliability for free, whereas the latter facilitates the use of the Twelvefold Way to increase the channel data rate and avoid decoding problems inherent in other network timing channels. We have implemented the Cloak encoder and decoder, and evaluated its goodput under controlled environment and in the wild. Moreover, we have in fact designed and evaluated a two-step detection algorithm for Cloak which, due to the space limit, could not be accommodated in this paper. Interested readers may refer to [20] for the algorithm and evaluation.

In a broader sense, our contribution in this work is to provide a new framework for designing more effective network covert channels. The ten encoding methods represent some of the design points in this framework. Based on this perspective, we should not rule out that there are other design points that possess other attractive properties. Therefore, one of the future work directions is to explore novel covert channel design with an even higher data rate than Cloak, for example. The other direction is on the detection aspect. Although the detection problem seems notoriously difficult, an active detection method is a promising approach. Another approach is to design more intelligent intermediaries that could reduce the channel data rate significantly.

Acknowledgment

The work described in this paper was partially supported by a grant from the Research Grant Council of the Hong Kong Special Administrative Region, China (Project No. PolyU 5080/02E) and a grant from the Cisco University Research Program Fund at Community Foundation Silicon Valley. The authors would like to thank the reviewers for their useful comments.

References

1. Shah, G., Molina, A., Blaze, M.: Keyboards and covert channels. In: Proc. USENIX Security (2006)
2. Singh, A., Nordstro, O., Lu, C., Santos, A.: Malicious ICMP tunneling: Defense against the vulnerability. In: Safavi-Naini, R., Seberry, J. (eds.) ACISP 2003. LNCS, vol. 2727, Springer, Heidelberg (2003)
3. Schechter, S., Smith, M.: Access for sale: A new class of worm. In: Proc. ACM Workshop on Rapid Malcode (WORM), ACM Press, New York (2003)
4. Rogers, R., Devost, M.: Hacking a Terror Network: The Silence Threat of Covert Channels. Syngress (2005)
5. Borders, K., Prakash, A.: Web Tap: Detecting covert Web traffic. In: Proc. ACM CCS, ACM Press, New York (2004)
6. Feamster, N., Balazinska, M., Harfst, G., Balakrishnan, H., Karger, D.: Infranet: Circumventing censorship and surveillance. In: Proc. USENIX Security (2002)
7. Wang, X., Reeves, D.: Robust correlation of encrypted attack traffic through stepping stones by watermarking the interpacket timing. In: Proc. ACM CCS, ACM Press, New York (2003)

8. Wang, X., Chen, S., Jajodia, S.: Tracking anonymous peer-to-peer VoIP calls on the Internet. In: Proc. ACM CCS, ACM Press, New York (2005)
9. Gligor, V.: A guide to understanding covert channel analysis of trusted systems (light pink book). Technical Report NCSC-TG-030, National Computer Security Center (1993)
10. DoD US: Department of defense trusted computer system evaluation criteria (orange book). Technical Report DoD 5200.28-STD, National Computer Security Center (1985)
11. Bishop, M.: Introduction to Computer Security. Addison-Wesley, Reading (2005)
12. Cabuk, S., Brodley, C., Shields, C.: IP covert timing channels: Design and detection. In: Proc. ACM CCS, ACM Press, New York (2004)
13. Watson, D., Smart, M., Malan, G., Jahanian, F.: Protocol scrubbing: Network security through transparent flow modification. In: IEEE/ACM Trans. Networking (2004)
14. Handley, M., Kreibich, C., Paxson, V.: Network intrusion detection: Evasion, traffic normalization, and end-to-end protocol semantics. In: Proc. USENIX Security Symp (2001)
15. Fisk, G., Fisk, M., Papadopoulos, C., Neil, J.: Eliminating steganography in Internet traffic with active wardens. In: Proc. Information Hiding Workshop (2002)
16. Stanley, R.: Enumerative Combinatorics. Cambridge University Press, Cambridge (1997)
17. Ahsan, K., Kundur, D.: Practical data hiding in TCP/IP. In: Proc. Workshop on Multimedia Security (2002)
18. Chakinala, R., Kumarasubramanian, A., Manokaran, R., Noubir, G., Rangan, C., Sundaram, R.: Steganographic communication in ordered channels. In: Proc. Information Hiding Workshop (2006)
19. Berk, V., Giani, A., Cybenko, G.: Detection of covert channel encoding in network packet delays. Technical Report TR2005536, Department of Computer Science, Dartmouth College (2005)
20. Luo, X., Chan, E., Chang, R.: Cloak: A ten-fold way for reliable covert communications (full version) (2007), `http://www.comp.polyu.edu.hk/~csrchang/CloakFull07.pdf`
21. Yeung, R.: A First Course in Information Theory. Kluwer Academic, Dordrecht (2002)