Robust Network Covert Communications Based on TCP and Enumerative Combinatorics

Xiapu Luo, *Member*, *IEEE*, Edmond W.W. Chan,
Peng Zhou, *Student Member*, *IEEE*, and Rocky K.C. Chang, *Member*, *IEEE*

Abstract—The problem of communicating covertly over the Internet has recently received considerable attention from both industry and academic communities. However, the previously proposed network covert channels are plagued by their unreliability and very low data rate. In this paper, we show through a new class of timing channels coined as Cloak that it is possible to devise a 100 percent reliable covert channel and yet offer a much higher data rate (up to an order of magnitude) than the existing timing channels. Cloak is novel in several aspects. First, Cloak uses the different combinations of N packets sent over X flows in each round to represent a message. The combinatorial nature of the encoding methods increases the channel capacity largely with (N, X). Second, based on the well-known 12-fold Way, Cloak offers 10 different encoding and decoding methods, each of which has a unique tradeoff among several important considerations, such as channel capacity and camouflage capability. Third, the packet transmissions modulated by Cloak can be carefully crafted to mimic normal TCP flows for evading detection. We have implemented Cloak and evaluated it in the PlanetLab and a controlled testbed. The results show that it is not uncommon for Cloak to have an order of channel goodput improvement over the IP Timing channel and JitterBug. Moreover, Cloak does not suffer from any message loss under various loss and reordering scenarios.

Index Terms—Network covert channel, timing channel, Enumerative Combinatorics, TCP, covert channel detection

1 Introduction

Hiding a communication channel between any two end points in the Internet and their messages therein is an important research problem due to its security and privacy ramifications. 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], [7], [8], [9], watermarking encrypted flows in stepping stones [10], tracking VoIP calls [11], and securing the transmission of syslog data [12].

Similar to the classic covert channels in trusted computer systems, network covert channels could be broadly classified into *storage channels* and *timing channels*. In a storage channel, the encoder and decoder communicate covertly through "attributes of shared resources" [13], which could be any fields in a packet that can be "written" by the encoder and "read" by the decoder. The covert messages are usually encoded directly into these fields. In a timing channel, they communicate "through a temporal or ordering relationship of accesses to a shared resource" [13] which could be the timing of packet arrivals that can be modulated by the encoder and observed by the decoder.

Existing network covert channels in literature suffer from low data rates in the presence of dynamic network conditions and *active network intermediaries* (ANIs) (i.e., protocol scrubbers [14], traffic normalizer [15], and active wardens [16]). For example, the message encoding based on *interpacket delay* (IPD) is very sensitive to delay jitter; a slight change in the network delay could cause decoding errors. Furthermore, packet losses affect the integrity of both timing and storage channels. Packet losses not only affect the decoding accuracy of individual messages, they could also destroy the framing structures, if any. On the other hand, the storage channels do not suffer from the negative effect of adverse network conditions. However, their encoded messages could be altered by an ANI which modifies the replaceable header fields in the packets that pass through it.

In this paper, we propose Cloak—a new class of timing channels which can achieve a 100 percent decoding accuracy, 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[1] data traffic as a cover (i.e., exploiting TCP's reliable transmission mechanism) and employing a fixed number (denoted by N) of TCP packets for encoding and decoding a message to avoid synchronization errors suffered by many network timing channels. Another important feature is that Cloak uses the different combinations of N packets sent over X flows in each round to encode a message. Due to the combinatorial nature of the encoding method, Cloak's channel capacity increases quickly with (N, X). Our extensive evaluation

• *The authors are with the Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Hong Kong.*
E-mail: {csxluo, cswwchan, cspzhouroc, csrchang}@comp.polyu.edu.hk.

Manuscript received 16 June 2011; revised 2 Apr. 2012; accepted 23 July 2012; published online 30 July 2012.
For information on obtaining reprints of this article, please send e-mail to: tdsc@computer.org, and reference IEEECS Log Number TDSC-2011-06-0153.
Digital Object Identifier no. 10.1109/TDSC.2012.64.

1. Other reliable transport protocols, such as SCTP, can also be used.

showed that Cloak can achieve an order of magnitude improvement in bit rate over prior work.

Besides, Cloak offers 10 different sets of encoding and decoding methods based on the classic 12-fold Way in Enumerative Combinatorics [17]. Each method makes different tradeoffs among several conflicting design goals, such as channel capacity and camouflage capability. To our best knowledge, Cloak is the *first* network covert channel that exploits Enumerative Combinatorics to convey hidden messages. Moreover, this approach can be applied to designing new covert channels and other steganography schemes. Although Cloak uses multiple flows to encode messages, the packet distribution over the flows can be carefully crafted to mimic the normal TCP behavior to evade detection.

For the rest of this paper, Section 2 first discusses the network covert channels proposed in the past that are relevant to Cloak and various countermeasures. Section 3 presents the basic idea of message encoding in Cloak. Section 4 details how we have resolved a number of difficult design issues for deploying Cloak in the Internet. Section 5 reports the PlanetLab (PL) measurement results of evaluating Cloak's data rate under various network conditions and parameter settings. Section 6 evaluates Cloak's capability to evade detection and discusses the tradeoff between undetectability and performance. Section 7 summarizes this paper with a few venues of enhancing this work.

2 Related Work

Due to page limit, we only review covert timing channels established in an Internet-wide environment. Interested readers may refer to [18] for other kinds of network covert channels, such as MAC layer timing channel and covert storage channel.

2.1 Covert Timing Channels

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 *interpacket delay channels*. In the class of ordered channels, Kundur and Ahsan proposed to resort the original order of a flow of IPSec packets and use the out-of-orderliness to imbed messages [19]. Chakinala et al. further extended the approach to TCP packets and formalized various models for these ordered channels [20]. El-Atawy and Al-Shaer rearranged packets according to the statistical distribution of packet ordering measured in the Internet [21]. However, the low packet reordering rate in the nowadays Internet limits the capacity of such timing channels because a warden could use packet reordering rate as a feature for detection. Moreover, some routers and firewall will rearrange reordered packets, thus rending such timing channels useless. Cloak does not reorder packets although permutation can further increase its capacity if it is allowed [22]. Instead, Cloak exploits the distribution of packets over multiple TCP flows for message encoding and therefore it has high capacity and better camouflage capability. Khan et al. suggested sending packets of different sizes on a set of TCP connections for transmitting covert messages [23]. It can be considered as a special case of Cloak with distinguishable packets and flows.

The class of IPD channels, on the other hand, embeds messages directly in the delay period between selected packets. Cabuk et al. [24] proposed an IP timing channel, where an IP packet arrival during a timing interval is decoded as 1 and the absence of it as 0. Shah et al. proposed JitterBug that encodes binary bits into the packet interarrival times without injecting new packets [1]. Berk et al. used IPD of ICMP packets to encode one or multiple bits [25]. For example, bit 1 is encoded by a longer IPD, whereas bit 0 is encoded by a smaller IPD. Cloak does not encode messages directly into the packet timing and thus can mimic normal packet timing for evading the detection and avoid the negative effect of network jitter and packet losses.

Since the IPDs carrying covert information may deviate from those in normal traffic [26], researchers recently proposed several methods to mimic the IPDs in normal traffic. Walls et al. designed Liquid that helps JitterBug evade the detection of entropy-based methods [26] by using a portion of IPDs to mitigate the shape distortions caused by JitterBug [27]. Gianvecchio et al. [28] first modeled the IPDs in normal traffic and then fitted the IPDs carrying hidden information into the model for normal IPDs. Sellke et al. [29] proved that by mimicking independent and identically distributed (i.i.d.) IPDs the timing covert channel can be made indistinguishable from normal traffic. Similarly, Liu et al. [30] proposed an undetectable covert timing channel for traffic with i.i.d. IPDs, which removes the assumption of bounded jitter in [29]. However, most of real traffic does not have i.i.d. IPDs and thus covert channels relying on i.i.d. assumption could be easily detected through features like IPDs' autocorrelation [31]. To mitigate the disturbance of covert message on IPDs' autocorrelation, Zander et al. suggested embedding covert message into the least significant parts of IPDs. However, its capacity is low and will be negatively affected by jitter. We proposed TCPScript that mimics TCP's bursty nature to send covert messages and can evade the detection based on IPDs' statistics [32]. TCPScript's capacity can be further improved by employing Cloak's combination methods. Since Cloak can mimic the normal IPD distribution (in Section 4.4) and flow statistics (in Section 6.2), it can evade not only the state-of-the-art detection schemes but also other possible detection methods.

Reliable transmission is important to covert channel for successfully transmitting messages. However, only a few studies examined the robustness of covert timing channels. Liu et al. employed the spread-spectrum technique to increase the Signal-to-Noise Ratio for improving the robustness of covert timing channels at the cost of capacity and camouflage capability [33]. Cloak cleverly exploits TCP's inherent reliable transmission service to provide robust transmission of covert message. We evaluate Cloak under various adverse network conditions in Section 5. To the best of our knowledge, Cloak is the first practical covert timing channel that can achieve *zero* error rate under such network conditions and Cloak does not need to sacrifice capacity and camouflage capability for reliable transmission.

Traffic watermarking schemes usually manipulate packets' timing information to embed the watermarks [34]. They are often used to track flows going through networks that hide the two ends of communications and introduce noticeable noise (e.g., Tor). Hence, existing traffic watermarking schemes just deliver 1-bit information (i.e., whether or not a flow carries the traffic watermark) and need additional mechanisms to increase the robustness [35]. Most of existing traffic watermarking schemes alter the IPD [36], [37], [38]. For example, RAINBOW enlarges or shortens individual IPDs independently [37]. Wang et al. proposed delaying a batch of packets within selected intervals [36], while SWIRL postpones packets to selected time intervals [38]. However, such traffic watermarks could be detected based on the anomalies they introduce to IPDs [39]. Cloak could evade such detection, because it does not manipulate the IPDs.

To embed stealthy traffic watermarks, Yu et al. manipulated the throughput of TCP flows according to pseudorandom noise (PN) codes used in the spread-spectrum communication [34]. However, this approach has to manipulate many packets [35] and the abnormal traffic pattern resulted from the PN codes could reveal this traffic watermark [40]. By exploiting Tor's protocol, Ling et al. proposed a novel approach to track flows through Tor by manipulating the amount of data cells strategically [35]. However, this attack relies on Tor's protocol. Different from these traffic watermarking schemes, Cloak provides high throughput, robustness, and stealthiness by employing the combinations of packets and flows to convey covert information, exploiting TCP's reliable transmission mechanism, and mimicking normal traffic's patterns. The techniques in Cloak could be used to improve the existing traffic watermarks.

2.2 Countermeasures

A few detection algorithms have been proposed for the IPD channels. They share the common thread that the detection is based on identifying anomalous statistics in IPDs. Cabuk et al. proposed anomaly detection methods for the IP timing channel based on the packet interarrival times [24]. Berk et al. proposed two methods for detecting the ICMP timing channels [25]. The first method is catered for a smart attacker who is assumed to possess the highest capacity. The second is for detecting binary channels similar to JitterBug. Based on the observation that covert channels may affect the distribution of IPD, Gianvecchio and Wang proposed two entropy-based methods to detect timing channels [26].

Besides detection, another defense mechanism is to mitigate the impact of a network timing channel. Some approaches employed for trusted computing systems could be extended to neutralize network timing channels. For example, Hu [41] and Giles and Hajek [42] suggested reducing the capacity of a timing channel in multilevel OS environment by introducing a long enough fuzzy time to each packet. The Network Pump assures that a Low net and a High net that are connected to the Network Pump will not compromise sensitive information. Although the Network Pump could not eliminate all timing channels, it could significantly suppress their throughput [43], [44], [45].

TABLE 1
An Example of Encoding a Hexadecimal Number Using Cloak with $(N, X) = (5, 3)$

Messages	n_1	n_2	n_3	Messages	n_1	n_2	n_3
0	5	0	0	8	2	1	2
1	4	1	0	9	2	0	3
2	4	0	1	10	1	4	0
3	3	2	0	11	1	3	1
4	3	1	1	12	1	2	2
5	3	0	2	13	1	1	3
6	2	3	0	14	1	0	4
7	2	2	1	15	0	5	0

3 Cloak

3.1 Encoding Based on Packet-Flow Distributions

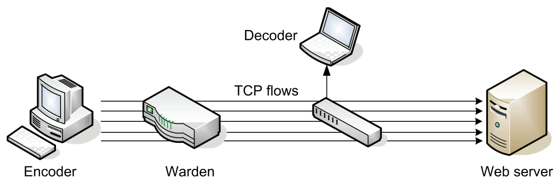
The covert messages in Cloak are encoded by a class of combinatorial objects—each covert message is encoded with a unique distribution of N TCP packets over X TCP flows. To defeat the repacketizing attack that a warden splits or merges packets to prevent the decoder from knowing the number of packets sent by the encoder, the encoder and decoder can replace packets with fix-sized chunks. For the ease of explanation, we still use packets when describing Cloak below.

The encoder and decoder agree on the values of N and X beforehand. Consider a simple example of $(N, X) = (5, 3)$. If the encoder and decoder could distinguish the three TCP flows, then there are a total of 21 possible ways of distributing the five packets over the three flows. Table 1 shows an example of encoding four-bit messages using 16 of the 21 combinations, where n_i is the number of packets sent on the ith flow. Furthermore, the encoder will transmit the next message only after receiving the acknowledgments (ACKs) for the N TCP packets. On the other side of the channel, the decoder starts decoding as soon as collecting N TCP packets from the encoder. As we will show in Section 4.1, the encoding and decoding can be performed using unranking and ranking functions (i.e., without a codebook).

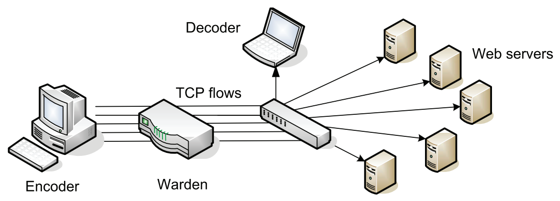
Cloak is reliable in the same sense of TCP reliability when messages experience packet losses, delay jitter, and other adverse conditions. First of all, Cloak's decoding accuracy is not affected by delay jitters, because the encoding is not based on the actual time. Second, since the encoder sends a 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 by retransmitting the unacknowledged TCP packets. 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 a normal TCP stack, no additional reliability mechanism is needed to guarantee Cloak's reliability.

3.2 Covert Communication Scenarios

In Fig. 1, we depict two possible 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. A passive warden attempts to detect network covert channels by analyzing all



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



(b) Five TCP flows connect to different Web servers.

Fig. 1. Two covert communication scenarios for Cloak.

the traffic sent between the encoder and any hosts outside the network. The passive detection does not alter the traffic flows and characteristics. An active warden, on the other hand, attempts to interfere with any network covert channels passing through it. The strategies usually involve altering the packet contents or the traffic characteristics, such as delaying and dropping packets.

In Fig. 1a, the encoder could establish a "normal" HTTP session with a remote server, consisting of five TCP flows. The encoder encodes the messages into the TCP flows, and the decoder eavesdrops at any point of the path and decodes the messages. In order to evade a passive warden, the encoder may distribute the TCP packets over the flows such that they match with the normal TCP behavior. 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, multithreaded upload or download (i.e., sending commands) has a similar traffic pattern.

In Fig. 1b, the encoder could establish normal HTTP sessions with multiple servers which are dispersed in different locations. Therefore, the decoder should be located on the common routing path for all servers. Although this approach restricts the decoder location, it can diffuse the relationship among the TCP flows. Moreover, it allows an encoder to select suitable flows from a large number of flows to different web servers for stealthily transmitting covert message. An approach for relaxing the location restriction is to use distributed information collection, for example, using a botnet where each bot observes partial information and sends it to the bot master.

3.3 The 12-Fold Way

Cloak offers 10 different encoding methods which are based on the well-known *12-fold Way* [17] in the field of Enumerative Combinatorics. The 12-fold Way refers to 12 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 $\mathbb{N}$ and the set of urns be $\mathbb{X}$, and $|\mathbb{N}| = N$ and $|\mathbb{X}| = X$ be their respective sizes. 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.

Table 2 summarizes the 12-fold Way using flows (urns) and packets (balls) [17], where $C_X^N = \frac{X!}{N!(X-N)!}$, $S(N, X) = \frac{1}{X!}\sum_{j=1}^{X}(-1)^{X-j}C_X^j j^N$, and $P(N, X)$ is the number of partitioning N into X parts. Each result 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 and therefore we have 10 encoding methods. In the rest of this paper, we refer the 10 encoding methods to as Cloak$^c(N, X)$ (or just Cloak^c if (N, X) is not necessary), $c \in [1, 10]$. According to Table 2, 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 clearly 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 flows (or packets).

3.4 The 10-Fold Way in Cloak

Cloak uses the 10 encoding methods to make tradeoffs between channel capacity and camouflage capability. By modeling a Cloak channel as a classic information channel, we can obtain the capacity of a Cloak$^c(N, X)$ channel in bits/symbol based on the mutual information [46]. Denoting the 12-fold Way result for Cloak$^c(N, X)$ by $T^c(N, X)$, a higher

TABLE 2
The 12-Fold Way and Their Relation to the 10 Encoding Methods ((1)-(10)) in Cloak

Elements of $\mathbb{N}$ (TCP packets)	Elements of $\mathbb{X}$ (TCP flows)	$\mathbf{f}_A$ (no restriction)	$\mathbf{f}_I$ (at most one packet in a flow)	$\mathbf{f}_S$ (at least one 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)

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 $1 \leq L \leq \lfloor \log_2 T^c(N, X) \rfloor$.

Flow and packet distinguishability generally increase channel capacity (e.g., $T^1(N, X) > T^7(N, X)$ for flow distinguishability and $T^1(N, X) > T^4(N, X)$ for packet distinguishability). Moreover, for each row in Table 2, the channel capacity for $\mathbf{f}_A$ is the largest (e.g., $T^1(N, X) > T^3(N, X)$, and $T^7(N, X) > T^8(N, X)$). However, there is a tradeoff between achieving a higher channel capacity by making the packets and flows distinguishable and camouflage capability. For the channels that require distinguishable packets (i.e., $c = 1, 3, 7, 8$), the encoder usually adds "markers" to the TCP packets in order to make them distinguishable. These additional markers may be noticed and even modified by a warden. Similar problems may occur also to the channels with flow distinguishability.

Based on the channel capacity, we define *data rate* in bits/second as $\frac{L}{T_s}$, 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 consider $N, X > 1$ in the rest of this paper.

4 Design Issues

4.1 Message Encoding and Decoding

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, e.g., stock indices [47]. The messages are encoded based on L-bit words, where $1 \leq L \leq \lfloor \log_2 T^c(N, X) \rfloor$. The encoder and decoder, however, do not need to exchange an explicit codebook. Two special functions, `Unrank()` and `Rank()`, are used instead for encoding and decoding, respectively. The `Rank()` function takes a packet-flow distribution and returns the index (starting from 0) of the array of all possible packet-flow distributions that are arranged in a lexicographically decreasing order. The index is known as *rank* in the field of Enumerative Combinatorics. The `Unrank()` function, on the other hand, performs the opposite. The message values in Table 1, for example, are the ranks for the first 16 packet-flow distributions.

Function 1. Unrank1(R, PFA[ ], X): integer[ ]

```
1: $Idx_{pkt} \leftarrow 0$;
2: while $R \neq 0$ do
3:     $Idx_{pfa} \leftarrow R \bmod X$;
4:     $PFA[Idx_{pfa}].add(Idx_{pkt})$;
5:     $R \leftarrow \frac{R - Idx_{pfa}}{X}$;
6:     $Idx_{pkt} \leftarrow Idx_{pkt} + 1$;
7: end while
8: return $PFA[\,]$;
```

Function 2. Rank1(PFA[ ], X): integer

```
1: $R \leftarrow 0$;
2: for $i = 0$ to $X - 1$ do
3:     for $j = 0$ to $|PFA[i]| - 1$ do
4:         $R \leftarrow X^{PFA[i].get(j)} \times i + R$;
5:     end for
6: end for
7: return $R$;
```

There are three major steps involved in sending a covert message. Each L-bit word is first converted to a nonnegative decimal value (through the `Bin2Dec()` function) that serves as the rank for the corresponding packet-flow distribution. The `Unrank()` function is then invoked to determine the distribution. Finally, the encoder marshals the packet-flow distribution 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 relies on TCP to recover them.

The three-step process above is exactly reversed for decoding a covert message. In the first step, the decoder unmarshals 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. 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 to 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 the `Rank()` function which yields the corresponding rank. As a last step, the rank is converted back to the L-bit word (through the function `Dec2Bin()`).

4.2 Ranking and Unranking Algorithms

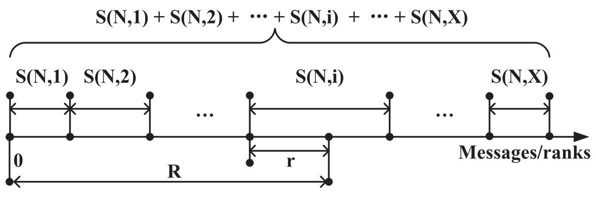
Both `Rank()` and `Unrank()` functions are required to be computationally efficient for a practical deployment of Cloak. A table-lookup approach will not work, because of the potentially huge packet-flow encoding space. Therefore, we instead design efficient ranking and unranking algorithms for the 10 encoding methods. Our first step is to design algorithms for the five primitive components in $T^c(N, X)$: X^N, C_X^N, $S(N, X)$, $P(N, X)$, and $\lambda!$, where $\lambda \in \{N, X\}$. There are already ranking and unranking functions for the last four components: C_X^N [48], $S(N, X)$ [49], $P(N, X)$ [48], and $\lambda!$ [50]. Therefore, we can directly apply them to $c = 4, 5, 6, 8, 10$.

For X^N which can be applied to $c = 1$, we notice that an integer R, where $0 \leq R \leq 2^L - 1$ and $L = \lfloor \log_2(X^N) \rfloor$, has a unique representation [51]:

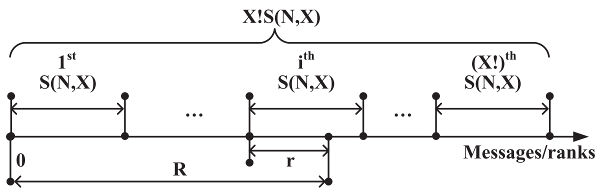
$$R = a_0 X^0 + a_1 X^1 + \cdots + a_{N-1} X^{N-1}, \quad a_i \in \{0, \ldots, X - 1\}. \tag{1}$$

To unrank a message R (which is the rank), we convert R into the form in (1) and then place a packet pkt_j, $0 \leq j < N$, into flow k, $0 \leq k < X$, if and only if $a_j = k$. The detailed algorithm is given in Function 1 in which Idx_{pkt} is the index of packet pkt, and $PFA[i]$ is the ith element in a packet-flow array containing the indices of the packets sent on flow i. To rank a given packet-flow distribution specified in $PFA[\cdot]$, we compute the right-hand side of (1). That is, the ranking algorithm shown in Function 2 basically converts a base X number into a nonnegative decimal number.

For the remaining methods (i.e., $c = 2, 3, 7, 9$), we propose a two-level approach to designing new ranking and unranking algorithms. We first consider $T^7(N, X)$ and $T^9(N, X)$, and use Cloak^7 as an example to illustrate our



(a) Cloak^7.



(b) Cloak^3.

Fig. 2. The messages (or ranks) for Cloak^3 and Cloak^7 are organized into two levels.

approach. Function 3 and Function 4 describe Cloak^7's unranking and ranking algorithms, respectively. Fig. 2a shows that the messages (or ranks) are organized into two levels with the first level composing of X subranges which have the number of messages given by the corresponding $S(N,i)$s. The second level includes the messages within each subrange according to $S(N,i)$'s unranking algorithm. Therefore, to encode a message R, Cloak^7's unranking algorithm first determines the number of flows (say i) by subtracting $\sum_{j=1}^{i-1} S(N,j)$ from R until the remaining value (say r) is not larger than $S(N,i)$ (in lines 2-6 of Function 3). Then, it determines how to allocate the N distinguishable packets into the i flows through $S(N,i)$'s unranking algorithm (`UnrankSNX`(R,N,i)) with input r (in line 7 of Function 3). `UnrankSNX`(R,N,i) is based on the algorithm `UnrankSetPtns` in [49]. It is worth noting that $X-i$ flows send nothing.

To decode the message after observing N packets, the decoder first restores r through $S(N,i)$'s ranking algorithm (`RankSNX`(PFA[ ], N, X) in line 1 of Function 4, and then obtains the original message $R = \sum_{j=1}^{i-1} S(N,j) + r$ (in lines 2-4 of Function 4). The algorithms for Cloak^9 can be constructed similarly. `RankSNX`(PFA[ ], N, X) is based on the algorithm `RankSetPtns` in [49].

Function 3. Unrank7(R, N, X): integer[ ]
```
1: $i \leftarrow 1$;
2: **while** $R > 0$ **do**
3:     $R \leftarrow R - S(N,i)$;
4:     $i \leftarrow i + 1$;
5: **end while**
6: $r \leftarrow R + S(N,i)$;
7: **return** UnrankSNX(r,N,i);
```

Function 4. Rank7(PFA[ ], N, X): integer
```
1: $R \leftarrow$ RankSNX($PFA$[ ], $N$, $X$); /* i.e., r */
2: **for** $i = 1$ to $X - 1$ **do**
3:     $R \leftarrow R + S(N,i)$;
4: **end for**
5: **return** $R$;
```

For $T^2(N,X)$ and $T^3(N,X)$, we use Cloak^3 as an example. Function 5 and Function 6 describe Cloak^3's unranking and ranking algorithms, respectively. Similar to before, Fig. 2b shows that the messages are also organized in two levels. The first level consists of $X!$ subranges, each of which has $S(N,i)$ number of messages. To encode a message R, Cloak^3's unranking algorithm first computes $r = R \; mod \; S(N,X)$ and $i = \frac{R-r}{S(N,X)}$ (in lines 1-2 of Function 5). The encoder then allocates the N distinguishable packets into the X flows using $S(N,X)$'s unranking algorithm (`UnrankSNX`(r,N,X)) with input r (in line 3 of Function 5). After that, the encoder permutes these flows according to a permutation of the X elements. This permutation is obtained through $\lambda!$'s unranking algorithm (`UnrankPermu`(i,X)) with input i (in line 4 of Function 5). We adopt the unranking algorithm in [50] to implement `UnrankPermu`(i,X) for its low-computational complexity. Finally, the encoder permutes the flows and then transmits these N distinguishable packets through the flows (in line 6 of Function 5).

For decoding, the decoder first computes i using $\lambda!$'s ranking algorithm (`RankPermu`(PFA,X)) and then calculates r through $S(N,X)$'s ranking algorithm (in lines 1-2 of Function 6). `RankPermu`(PFA,X) is based on the ranking algorithm in [50]. Finally, it restores the original message $R = iS(N,X) + r$. The algorithms for Cloak^2 can be constructed similarly.

4.3 A Head-of-Line Blocking Problem (HoLB)

We tackle in this section a head-of-line blocking problem that we encountered during our Internet experimentation. 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 several RTTs to complete the transmissions for the busy flow, thus leading to a low data rate. The problem may worsen if there are packet losses in the most busy flow. This problem will also occur to the flows that connect to different servers with various RTTs.

Function 5. Unrank3(R, N, X): integer[ ]
```
1: $r \leftarrow R$ mod $S(N,X)$;
2: $i \leftarrow \frac{R-r}{S(N,X)}$;
3: $PFA \leftarrow$ UnrankSNX($r$,$N$,$X$);
4: $PMU \leftarrow$ UnrankPermu($i$,$X$);
5: **return** Permutate($PRA$,$PMU$);
```

Function 6. Rank3(PFA[ ], N, X): integer
```
1: $i \leftarrow$RankPermu($PFA$,$X$);
2: $r \leftarrow$RankSNX($PFA$,$N$,$X$);
3: **return** $i \times S(N,X) + r$;
```

We propose two solutions to tackle the HoLB problem: a new *D-limited codeword scheme* (in Section 4.3.1) and an aggressive transmission scheme (in Section 4.3.2). The D-limited codeword scheme cleverly selects codewords that will not lead to the HoLB problem. The aggressive scheme, on the other hand, sends all N packets at the same time without considering cwnd and aggressively retransmits any lost packets. Therefore, this method can admit all codewords at the expense of deviating from the normal TCP behavior.

4.3.1 A D-Limited Codeword Scheme

The D-limited codeword scheme essentially limits 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 parameter D should be 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 by first defining the following two quantities:

1. Let $\Upsilon(N, D)$ 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 to 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., $c = 10$).

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

Proof. Please find the proof in [52]. □

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

Proof. Please find the proof in [52]. □

Proposition 1 computes how much information this D-limited codeword scheme could transmit. Corollary 1 further shows that if the upper bound of flows is X, then $N \leq X + D - 1$. 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 the conjugate of ζ [17], denoted by ζ', and transmits the 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}'s ranking algorithm to decode the message.

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

4.3.2 Aggressive Transmission Scheme

Unlike the D-limited codeword scheme that uses a subset of all codewords, the aggressive transmission scheme can admit all codewords. The basic idea is that the encoder will dispatch all packets belonging to the kth message after receiving ACKs 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 ACKs before T_E, it will retransmit the unacknowledged packets and reset the timer. Similar to the one used in normal TCP, T_E is usually set to the estimated RTT that can be obtained through an exponentially weighted moving average of RTT samples.

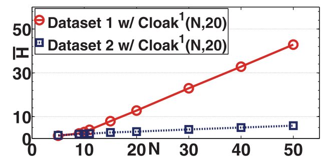
4.4 Detection Evasion

To evade the detection algorithms designed for timing channels [24], [26], Cloak can mimic the behavior of normal HTTP flows. Consider that a Cloak encoder obtains the traffic statistics of normal TCP flows. It records, for example, the number of data packets, the sequence of IPD, and the RTT for each flow. Before sending n_i packets in the ith flow, the encoder selects a flow record that sends more data packets than n_i and has an RTT no less than the RTT for the covert channel. After identifying the flow record, the encoder sends out the packets according to the IPDs in the selected flow record. In Section 6.1, we apply this approach to evaluate Cloak's undetectability. Moreover, according to our experience [53], the partial ACK approach is promising for making packets distinguishable. Based on our measurement for more than 100 popular security websites, the partial ACKs can traverse the firewalls in all cases. On the other hand, using the receive window size and packet length for the markers runs into a much higher risk of being detected, because their distributions deviate from the normal usages. We discuss how to evade detections based on flow characteristics in Section 6.2.

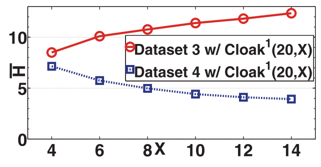
5 Performance Evaluation

We have implemented Cloak, the IP timing channel (IPTime in short) [24] and JitterBug (JBug in short) [1], and conducted extensive evaluation on the PL platform and a controlled testbed. Due to page limit, we only present the PL results here and leave the testbed results and the implementation details to [52]. Cloak has two versions: one uses TCP socket (SOCK_STREAM) and the other employs raw socket (SOCK_RAW).

Since Cloak is a reliable covert channel, we concentrate on evaluating its data rate which is measured in terms of the empirical channel goodput: $G = (1 - B_e)\frac{NL}{T_d}$, where T_d is the total time required for delivering N L-bit covert messages, and B_e is the channel's bit error rate (BER) which is computed based on the Levenshtein distance. Since Cloak can automatically recover errors, its $B_e = 0$. However, the BERs of other timing channels are not always 0. Moreover,



(a) Datasets 1 and 2 under different values of N.



(b) Datasets 3 and 4 under different values of X.

Fig. 3. The average degrees of HoLB for four data sets of packet-flow codewords for PlanetLab experiments.

wherever we find appropriate, we will compare Cloak with IPTime and JBug based on channel goodput and BER.

We have conducted Internet experiments on nine PL nodes in China (CN), Japan (JP), California (CA), Kansas (KS), Rhode Island (RI), Korea (KR), Belgium (BE), United Kingdom (UK), and Portugal (PT). Each PL node, serving as a Cloak encoder, initiates network covert channels with a decoder which, together with a web server, is deployed in our campus network in Hong Kong. The encoder embeds covert messages into the TCP packets sent to the web server, and the decoder receives the messages by eavesdropping the TCP packets. It follows the communication scenario shown in Fig. 1a. We will evaluate another scenario shown in Fig. 1b in the future work.

5.1 Impacts of HoLB on Channel Performance

To evaluate the impact of the HoLB problem, we define the *degree of HoLB* of a message by $H = \max_{0 \le i < X} n_i$. As H increases, the encoder is expected to take a longer time to transmit the message. We considered Cloak$^1(N, 20)$ with $N = 5, 9, 10, 11, 15, 20, 30, 40, 50$, and for each $(N,\ X)$ tuple, we generated two data sets (data sets 1 and 2) of 100 L-bit words. In data set 1, we purposely assigned more packets to flow 1; in data set 2, each packet was assigned to each flow with the same probability. As a result, Fig. 3a shows that $\overline{H}$ for data set 1 increases 10 times faster than that for data set 2 after N is increased beyond 10. We have also considered Cloak$^1(20, X)$ with $X = 4, 6, 8, 10, 12, 14$ and generated another two data sets (data sets 3 and 4) similarly as data sets 1 and 2. Fig. 3b shows that the two graphs diverge as X increases.

We used Cloak encoder (SOCK_STREAM) and the four data sets of codewords. Each encoder delivered each set of codewords for 30 times based on which we computed the average channel goodput. Fig. 4 reports average goodput $\overline{G}$ for the four data sets. For each N, we sorted the results in the ascending order of the measured mean RTTs. The Cloak encoder at CN, which has the least RTTs, always achieves higher goodputs than other PL nodes. The maximum channel goodput obtained by CN is around 450 bit/s. The plots also show that for a given N, the channel goodput generally tends to decrease with the mean RTT.

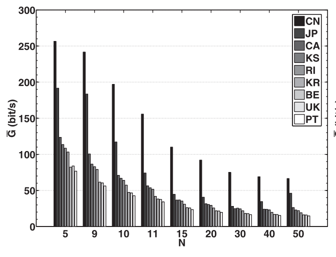
The channel goodputs for the two load-balanced cases (i.e., data sets 2 and 4) increase with N and X for all PL nodes. This shows that when the packet-flow distributions are balanced, an increase in either N or X will increase $\overline{G}$. On the other hand, an opposite trend is shown in Fig. 4a for data set 1. Since data set 1 represents a highly unbalanced loading, increasing N will further aggravate the HoLB problem, causing a decline in the channel goodput. Although such a decline cannot be observed in Fig. 4c for data set 3, the channel goodputs actually peak at $X = 10$ or 12, and they slightly decline after that.

5.2 Evaluation of the D-Limited Codeword Scheme

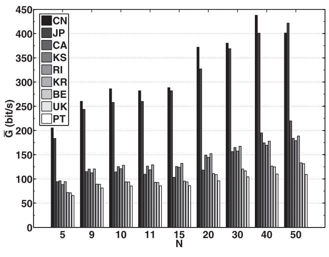
To measure the performance of the D-limited codeword scheme, we selected five PL nodes (JP, CA, KS, KR, and BE) and measured their channel goodput. Similar as before, we generated a set of 100 L-bit binary codewords for each $(N, 6)$ tuple, where $N = 12, 16, 20$. Each Cloak$^1(N, X)$ encoder encoded them into two distinct sets of packet-flow codewords: one generated by the D-limited codewords scheme with $D = 6$ and the other by the normal encoding scheme. Since the goal is to study the raw performance gain achieved by the D-limited codewords, we use only the SOCK_STREAM implementation. For each set of codewords, we compute $\overline{G}$ based on 30 independent runs.

Fig. 5 shows $\overline{G}$ for the five PL nodes. The figure shows that the D-limited scheme always gives a higher $\overline{G}$ than the normal scheme for all nodes and for all three (N, X) tuples. In particular, we note a maximum gain of 77 percent from the JP node with Cloak$^1(20, 6)$ and a minimum gain of 1.6 percent by the KR node with Cloak$^1(12, 6)$. Moreover, some of the nodes, such as KS and KR, attain less performance gain than other nodes. By examining the traffic traces, we have found that the packet loss rates at these two nodes were much lower than the other nodes. Therefore, the normal scheme has already provided a very high $\overline{G}$, thus eroding the benefit of adopting the D-limited scheme.

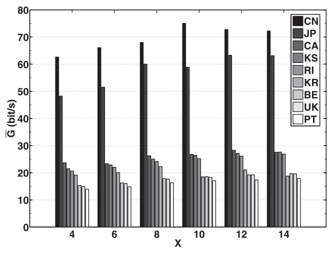
We also evaluated the aggressive transmission scheme in [52] and found that it can achieve much more improvement in the channel goodput than the D-limited codeword scheme. However, the aggressive transmission scheme may be vulnerable to the detection, because its traffic volume could be intensive, and the traffic pattern does not conform with normal TCP traffic pattern. Therefore, after taking the detection risk into consideration, the D-limited codeword scheme would still be a better solution to mitigating the impact of the HoLB problem.



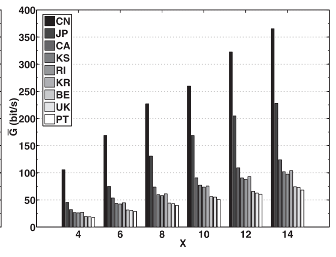
(a) Dataset 1.



(b) Dataset 2.

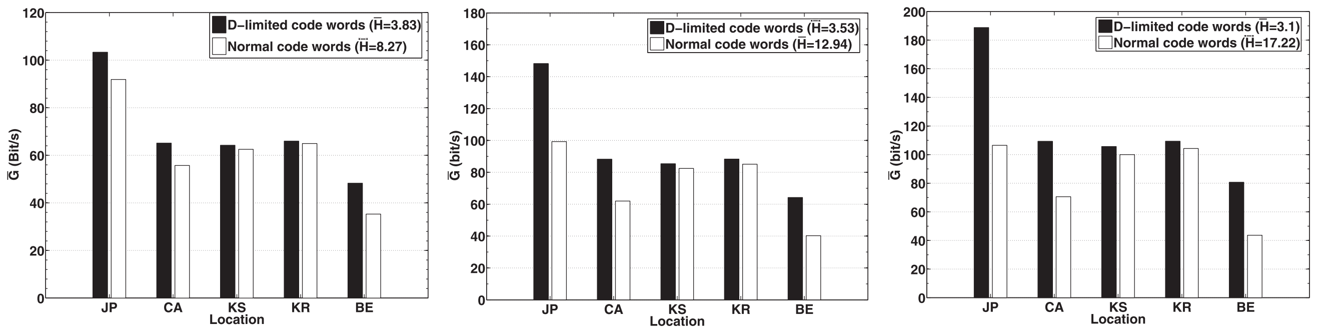


(c) Dataset 3.



(d) Dataset 4.

Fig. 4. Channel goodput for transmitting four data sets of 100 codewords using the SOCK_STREAM implementation of Cloak$^1(N, X)$ encoder in the PlanetLab.



(a) Cloak$^1(12, 6)$. (b) Cloak$^1(16, 6)$. (c) Cloak$^1(20, 6)$.

Fig. 5. A comparison of the average channel goodputs obtained by the D-limited codewords and normal codewords for five PlanetLab nodes using the `SOCK_STREAM` implementation of Cloak encoder with $D = 6$.

5.3 Comparing Cloak, IPTime, and JBug

We compare Cloak, IPTime, and JBug using the five PL nodes. In this set of experiments, we generated another 100 packet-flow codewords using the normal Cloak$^1(20, 4)$ encoder with $\overline{H} = 5.86$. Each node used both the `SOCK_RAW` and `SOCK_STREAM` implementations to transmit the codewords. Similar to the previous experiments, we set T_E to the corresponding measured mean RTT for the Cloak encoder. For JBug and IPTime, the encoder marshalled each respective binary codeword directly into a flow of modulated UDP packets with the time interval $w = \mathrm{RTT}, 1.5\ \mathrm{RTT}$ [52]. Both $\overline{G}$ and BER are computed based on 30 independent runs.

Table 3 reports the results, in which the two leftmost values in each cell correspond to the lower limit and upper limit of the 95 percent confidence intervals for $\overline{G}$, and the value inside the parentheses corresponds to the measured BER. The table clearly shows that the two versions of Cloak encoder achieve much better performance than JBug and IPTime. In particular, the Cloak (`SOCK_RAW`) encoder at the JP node attains a throughput of more than 200 bit/s. Comparatively, JBug and IPTime could generate no more than 14 and 10 bit/s, respectively. Besides, both Cloak channels provide error-free channels, but JBug and IPTime suffer from decoding inaccuracy, with the largest measured BERs of 2.65 and 3.63 percent. Furthermore, we calculate the BERs using the Hamming distance for each channel (and the detailed results are not shown here), and the average BERs for JBug and IPTime are as high as 49 percent even when using a larger $w = 1.5$ RTT.

6 Evaluating Cloak's Undetectability

6.1 Detection Based on Interpacket Delay

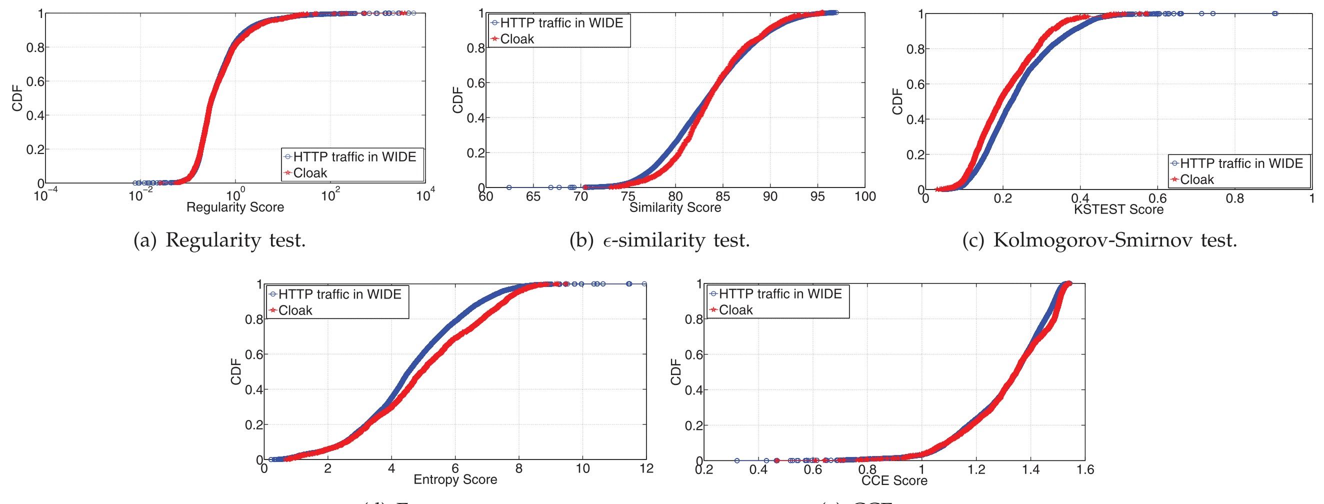
Since Cloak is a class of timing channels, we first evaluate Cloak's undetectability by applying the state-of-the-art methods for detecting timing channels to Cloak traces. We have implemented the Regularity test [24], ϵ-similarity test [24], Kolmogorov-Smirnov (KS) test [28], EN test [26], and CCE test [26] for the evaluation. Using the same parameter settings suggested in the original papers, we set the nonoverlapping window of size ω utilized in the Regularity test to 100 [24] and ϵ used by the ϵ-similarity test to 0.005 [24]. Following the approach in [28] for the KS test, we have created the normal distribution of the IPDs for a set of 779,913 IPDs which were obtained from 10,000 HTTP flows randomly selected from the WIDE data set [54]. The WIDE data set contains *all* traffic going through its sample point-F during March 30, 2009-April 2, 2009, and the size of packet header traces is around 433 GB.

We randomly selected 8,000 HTTP flows, each of which contains at least 2,000 IPDs following the suggestion in [26], from the WIDE data set as normal HTTP traffic. Cloak mimicked the normal IPDs in 1,500 HTTP flows randomly selected from the 8,000 flows based on the method described in Section 4.4. We applied the five detection methods to both normal HTTP traffic and Cloak traffic. Since each method will first compute a score for a flow, we plot the CDFs of these scores generated by the methods in Fig. 6. Each subfigure contains two CDF curves, one for the normal traffic and the other for the Cloak traffic. Fig. 6 shows that Cloak can successfully mimic the normal traffic, because the two CDFs are very close to each other in each test.

The score is used for determining whether a flow carries a covert channel. For example, the Regularity test [24] classifies a flow as a covert channel if it has a score smaller than a threshold. The ϵ-similarity test regards a flow as a covert channel if its score is larger than a threshold [24]. For the KS test, a flow is labeled as a covert channel if its KS test score is larger than a threshold [26], [28]. In the entropy-based methods, the distribution of a flow's IPDs is compared with that for normal traffic to detect timing channels. Following the suggestions in the original papers,

TABLE 3
Channel Goodputs Obtained by Cloak, IPTime, and JBug from Five PlanetLab Nodes

Loc.	Lower bound of a 95% confidence interval of G / upper bound of a 95% confidence interval of G (BER)					
	Cloak (`SOCK_RAW`)	Cloak (`SOCK_STREAM`)	JBug (w =RTT)	JBug ($w = 1.5$ RTT)	IPTime (w = RTT)	IPTime (w = 1.5RTT)
JP	203.9/216.3 (0)	55.2/58.6 (0)	13.0/13.0 (.016)	8.8/8.8 (.016)	9.5/9.5 (.036)	6.5/6.5 (.021)
CA	85.7/88.5 (0)	68.8/71.8 (0)	7.3/7.4 (.027)	4.8/4.8 (.052)	5.4/5.5 (.028)	3.7/3.7 (.016)
KS	90.8/91.0 (0)	66.6/69.3 (0)	6.1/6.1 (.002)	4.1/4.1 (.001)	4.5/4.5 (.011)	3.0/3.0 (.009)
KR	106.5/107.9 (0)	68.7/71.5 (0)	5.7/5.7 (.001)	3.8/3.8 (.001)	4.2/4.2 (.013)	2.8/2.8 (.008)
BE	63.9/64.2 (0)	44.7/45.6 (0)	4.4/4.4 (.001)	2.9/2.9 (.001)	3.2/3.2 (.008)	2.1/2.2 (.007)



(a) Regularity test. (b) ϵ-similarity test. (c) Kolmogorov-Smirnov test.

(d) Entropy test. (e) CCE test.

Fig. 6. CDFs of normal traffic's scores and the Cloak traffic's scores generated by five detection methods.

we select the threshold for each detection algorithm by setting its false positive rate to no more than 0.01. Table 4 shows that the detection rates for Cloak are very low (less than 0.04), supporting that Cloak can effectively evade these detection methods. These results therefore suggest that the current methods designed for detecting IPD-based timing channels are not sufficient for Cloak which does not directly manipulate the IPDs for message encoding.

6.2 Detection Based on Flow Characteristics

Although the methods for detecting timing channels are not sufficient for Cloak, other traffic characteristics may be disturbed by the Cloak encoding methods. In particular, we consider flow size, flow size distribution, and burst size for TCP flows. A TCP flow generated from a web client is usually short, because it normally carries only HTTP requests. As a result, using a fixed set of flows to deliver many covert messages continuously will lead to long TCP flows, which are suspicious for those initiated by normal clients.

Besides the flow size, the flow size distribution could be used to detect a set of TCP flows modulated by a Cloak encoder. If the covert messages are randomly selected, the numbers of packets sent onto individual flows would be similar, resulting in a uniform distribution of flow sizes. Although the actual flow size distribution depends on the applications, it is hardly uniform. Therefore, a uniform flow-size distribution will flag the set of flows as anomalous.

The burst size of a TCP flow refers to the number of packets sent out together. For a bulk transfer, the burst size

TABLE 4
Per-Flow Detection Rates for Cloak Using Five Detection Methods for IPD-Based Timing Channels

Detection methods	Positive if the scores	Normal HTTP's false positives	Cloak's detection rates
Regularity	≤ 0.0736	0.01	0
ϵ-similarity	$\geq 94.55\%$	0.01	0.01
KSTEST	≥ 0.4869	0.01	0
Entropy	≤ 0.8266	0.01	0
CCE	≥ 1.5181	0.01	0.04

will increase with the send window size, if no retransmission event occurs. However, a Cloak encoder using the D-limited codeword will send out at most D packets in a burst, and the burst size is also uniformly distributed between 1 and D. Thus, the burst size can be used to detect Cloak channels.

To help Cloak evade detection based on the three anomalous flow characteristics, a Cloak encoder first sends a single L-bit word on a TCP flow. Since the packets are sent through the default TCP/IP stack for the `SOCK_STREAM` implementation, the burst size will follow the normal behavior. For the flow size and flow size distribution, the encoding process is based on a given set of normal TCP flow sizes, denoted by $\mathbb{S}$, which Cloak tries to mimic. The main idea is to use *only* these legitimate flow sizes for encoding. Therefore, we need a mapping between n_i (where $n_i \leq N$) for the ith flow and a legitimate flow size. A simple, and yet effective, way to achieve it is to use mod $(N+1)$ to map a flow size to n_i. The details are given below:

1. Each flow size $s \in \mathbb{S}$ is first mapped to $[0, 1, \dots, N]$ by performing $s \bmod (N+1)$.
2. To transmit an L-bit word, the encoder first converts it into $n_1, n_2, \dots, n_X$ (i.e., the distribution of N packets over X flows) using Cloak$^c(N, X)$, and then selects the flow size for the ith flow by $n_i = s \bmod (N+1)$ for some $s \in \mathbb{S}$.
3. For the ith flow, the n_i packets are then marshalled in a flow with the flow size selected from step (2).
4. After receiving the ith flow, the decoder first obtains n_i by performing $\bmod (N+1)$ on the flow size. The message is then decoded according to Cloak's decoding procedure.

In step 2 above, it is possible that some n_i may not be mapped to any flow size. With a sufficient number of flow sizes in $\mathbb{S}$, this problem can be resolved by reducing N. That is, the choice of N is now subject to the constraint of $\mathbb{S}$. On the other hand, if there are multiple flow sizes that can be used for n_i, the encoder selects one of them according to their relative probabilities.

Lastly, it is important to point out that this flow-size-based encoding method can also defeat the repacketizing attack

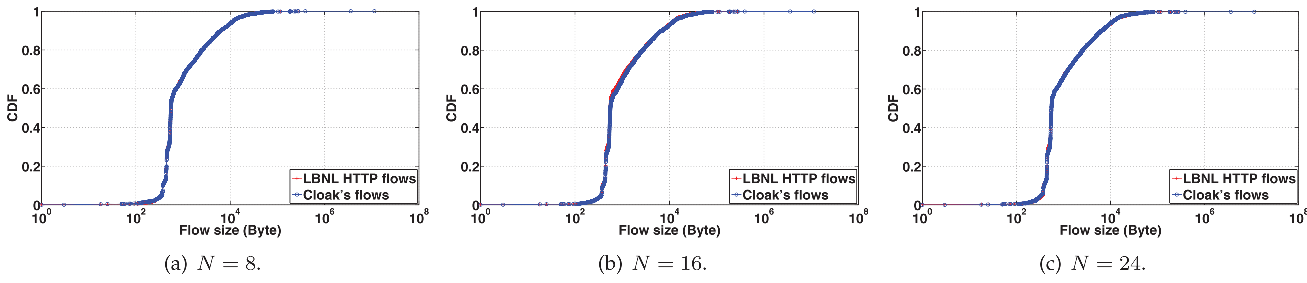


Fig. 7. CDFs of HTTP's flow sizes in the LBNL's trace and those of Cloak's flow sizes.

discussed in Section 3, because the messages are not encoded and decoded directly based on the number of packets.

We have evaluated this new method using the LBNL's enterprise traces, which contain more than 100 hours of packet traces collected from an enterprise network having several thousand hosts [55]. We selected the HTTP request flows with destination ports equal to 80. Fig. 7 shows the CDF of their flow sizes. Figs. 7a, 7b, and 7c also show the CDFs of Cloak's flows sending 5,000 random integers in the range of $[0, N]$. The results show that Cloak can successfully mimic normal flows, thus circumventing detection based on the three anomalous flow characteristics. We also instructed Firefox with default setting to automatically visit the top 2,000 websites ranked by Alexa, Inc., and captured the request flows. The results show that Cloak can also mimic such flows successfully. Due to page limit, all results related to the Alexa data set are presented in [52].

We have evaluated the goodput of the new method on five PL nodes. Due to page limit, we report the results obtained from a PL node in Japan and leave other results to [52]. In the experiment, a host in Hong Kong used the new method to deliver 5,000 random L-bit words to each PL node. Fig. 8 shows the goodput's CDF when the new method mimicked flows in the LBNL data set and employed Cloak$^1(N, X)$ with $X = 3, 6$ and $N = 8, 16, 24$.

Fig. 8 shows three interesting observations. First, the goodput spreads in a wide range. For example, when $X = 3$ and $N = 24$, the goodput ranges from 1.36 to 198.48 bit/sec. This result is due to the wide flow size distribution in the LBNL data set. If n_i is mapped to a short flow, the goodput is high. Otherwise, the goodput is low. Second, given an X, a larger N leads to higher goodput, because $T^1(N, X)$ increases with N while the flows are selected from the same training data set. Similarly, a larger X results in higher goodput, because $T^1(N, X)$ increases with X and all flows can be established simultaneously to avoid additional delay.

Third, by comparing Fig. 8b with Fig. 5b, we found that under the same configuration (i.e., $N = 16$ and $X = 6$) the new method's goodput is higher than the average (i.e., 100 bits/sec for normal codewords in Fig. 5b) in around 90 percent of the cases. It can also achieve higher goodput than the average of the D-limited codewords in around 30 percent of the cases. It is because most of the LBNL flows are short and the packet-based Cloak sets each packet's payload to 1,460 bytes. Fig. 9 shows the maximal amount of bytes sent by individual flows for each L-bit word. For the packet-based Cloak, this value equals to $\max\{n_i\} \times 1{,}460$. The graphs with labels "$N = 16$ D-limited" and "$N = 16$" indicate the vanilla Cloak with and without using D-limited codeword, respectively. For the flow-size-based Cloak, Fig. 9 shows the longest flow's size for each L-bit word. The graphs with labels "Alexa" and "LBNL" denote the flow-size-based Cloak mimicking flows from the LBNL and Alexa data sets, respectively. We can see that the packet-based Cloak usually sends a longer flow than the new method does, thus having a lower goodput. However, the new method's goodput will decrease if the training flows are long and the packet-based method can use a smaller packet size to increase the goodput.

Although the flow-size-based method can evade the detection based on flow characteristics by mimicking the normal flow size, the user needs to carefully prepare the packet payload and schedule the TCP connections, because the order of HTTP flows may contain semantic information. We will investigate such detection method and the countermeasures in future work.

6.3 Tradeoff between Undetectability and Capacity

Clearly, there are tradeoffs between undetectability and capacity for Cloak. Section 3.4 shows that either packet or flow distinguishability will increase the capacity, and we have discussed methods for making them distinguishable in [52]. However, distinguishability may increase the risk of being detected if the object that makes packet or flow distinguishable is regarded as abnormal by a warden, for example, using the receiving window field in the TCP header. Therefore, if it is safe to let packets and flows

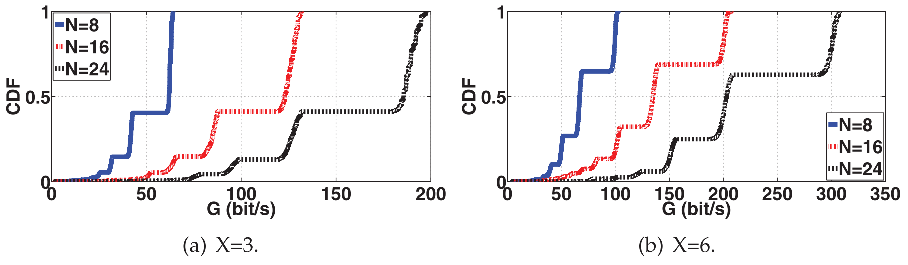


Fig. 8. Goodput of the flow-size-based encoding method mimicking flows in the LBNL data set.

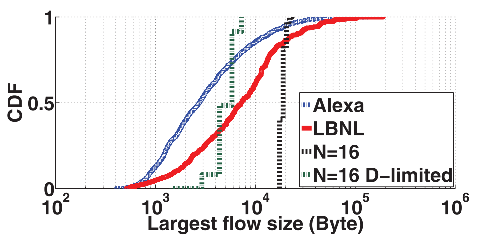


Fig. 9. The maximal amount of bytes sent by individual flows for each L-bit word.

distinguishable, the encoder can adopt encoding methods 1, 2, and 3. However, if only packets (or flows) should be made distinguishable, the encoder can choose methods 7 and 8 (or 4, 5, and 6). Otherwise, methods 9 and 10 can be used to evade existing detection schemes that look for abnormal IPDs and special markers at the cost of lower capacity.

Section 6.2 proposes a flow-size-based method to evade detection based on three anomalous flow characteristics. However, it may reduce Cloak's capacity. For example, since each TCP flow can send only a single L-bit word, the capacity will be low if the training flows are long. Moreover, if the TCP connections have to be established one by one after some delays to evade potential detection, the new method may introduce significant delay due to the frequent establishment and disconnection. As part of the future work, we will investigate how to encode more covert messages into a set of flows, while keeping the flow size distribution similar to a normal one. Since scrutinizing sophisticated features in all flows will increase a warden's overheads significantly, the tradeoff among channel capacity, undetectability, and a warden's burden will be another interesting issue to investigate in future work.

7 Conclusions and Future Work

In this paper, we proposed 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 reliability for free, while the latter facilitates the use of the 12-fold Way to increase the channel data rate and avoid general decoding problems in network timing channels. We implemented and evaluated Cloak under controlled environment and in the wild.

In a broader sense, our contribution in this work is to provide a new framework for designing more effective network covert channels. The 10 encoding methods represent some of the basic design points in this framework. Based on this perspective, we should not rule out other design points that could possess attractive properties. Therefore, one of the future work directions is to explore novel covert channel design with an even higher data rate than Cloak. The other direction is on the detection aspect. Although the detection problem seems notoriously difficult, an active detection method is a promising approach. For example, a warden may intentionally disturb a few suspicious flows and then monitor any anomaly within a group of suspicious flows. Another approach is to design more intelligent intermediaries that could reduce the channel data rate significantly. For instance, as an HTTP proxy, the intermediary may establish a random number of flows and then dispatch users' requests to different flows.

Acknowledgments

The authors thank the reviewers for their careful reading of the manuscript and for their very helpful suggestions. This work is partially supported by a grant G-U386 from The Hong Kong Polytechnic University and a grant (60903185) from NSFC.

References

[1] G. Shah, A. Molina, and M. Blaze, "Keyboards and Covert Channels," *Proc. 15th USENIX Conf. Security Symp.* , 2006.
[2] A. Singh, O. Nordstro, C. Lu, and A. Santos, "Malicious ICMP Tunneling: Defense against the Vulnerability," *Proc. Australasian Conf. Information Security and Privacy*, 2003.
[3] S. Schechter and M. Smith, "Access for Sale: A New Class of Worm," *Proc. ACM Workshop Rapid Malcode (WORM)*, 2003.
[4] R. Rogers and M. Devost, *Hacking a Terror Network: The Silence Threat of Covert Channels.* Syngress, 2005.
[5] M. Bauer, "New Covert Channels in HTTP: Adding Unwitting Web Browsers to Anonymity Sets," *Proc. ACM Workshop Privacy in the Electronic Soc.*, 2003.
[6] K. Borders and A. Prakash, "Web Tap: Detecting Covert Web Traffic," *Proc. 11th ACM Conf. Computer and Comm. Security (CCS)*, 2004.
[7] N. Feamster, M. Balazinska, G. Harfst, H. Balakrishnan, and D. Karger, "Infranet: Circumventing Censorship and Surveillance," *Proc. 11th USENIX Security Symp.*, 2002.
[8] N. Feamster, M. Balazinska, W. Wang, H. Balakrishnan, and D. Karger, "Thwarting Web Cenorship with Untrusted Messenger Discovery," *Proc. Privacy Enhancing Technologies (PET) Workshop*, 2003.
[9] S. Burnett, N. Feamster, and S. Vempala, "Chipping Away at Censorship with User-Generated Content," *Proc. USENIX Security Symp.*, 2010.
[10] X. Wang and D. Reeves, "Robust Correlation of Encrypted Attack Traffic through Stepping Stones by Watermarking the Interpacket Timing," *Proc. 10th ACM Conf. Computer and Comm. Security (CCS)*, 2003.
[11] X. Wang, S. Chen, and S. Jajodia, "Tracking Anonymous Peer-to-Peer VoIP Calls on the Internet," *Proc. 12th ACM Conf. Computer and Comm. Security (CCS)*, 2005.
[12] D. Forte, C. Maruti, M. Vetturi, and M. Zambelli, "SecSyslog: An Approach to Secure Logging Based on Covert Channels," *Proc. First Int'l Workshop Systematic Approaches to Digital Forensic Eng. (SADFE)*, 2005.
[13] M. Bishop, *Introduction to Computer Security.* Addison-Wesley, 2005.
[14] D. Watson, M. Smart, G. Malan, and F. Jahanian, "Protocol Scrubbing: Network Security through Transparent Flow Modification," *IEEE/ACM Trans. Networking*, vol. 12, no. 2, pp. 261-273, Apr. 2004.
[15] M. Handley, C. Kreibich, and V. Paxson, "Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics," *Proc. USENIX Security Symp.*, 2001.
[16] G. Fisk, M. Fisk, C. Papadopoulos, and J. Neil, "Eliminating Steganography in Internet Traffic with Active Wardens," *Proc. Information Hiding Workshop*, 2002.
[17] R. Stanley, *Enumerative Combinatorics.* Cambridge Univ. Press, 1997.
[18] S. Zander, G. Armitage, and P. Branch, "A survey of Covert Channels and Countermeasures in Computer Network Protocols," *IEEE Comm. Surveys and Tutorials*, vol. 9, no. 3, pp. 44-57, Third Quarter 2007.
[19] K. Ahsan and D. Kundur, "Practical Data Hiding in TCP/IP," *Proc. Workshop Multimedia Security*, 2002.
[20] R. Chakinala, A. Kumarasubramanian, R. Manokaran, G. Noubir, C. Pandu Rangan, and R. Sundaram, "Steganographic Communication in Ordered Channels," *Proc. Eighth Int'l Conf. Information Hiding*, pp. 42-57, 2007.

[21] A. El-Atawy and E. Al-Shaer, "Building Covert Channels over the Packet Reordering Phenomenon," *Proc. IEEE INFOCOM,* 2009.

[22] X. Luo, P. Zhou, E. Chan, R. Chang, and W. Lee, "A Combinatorial Approach to Network Covert Communications with Applications in Web Leaks," *Proc. IEEE/IFIP 41st Int'l Conf. Dependable Systems & Networks (DSN),* 2011.

[23] H. Khan, Y. Javed, S. Khayam, and F. Mirza, "Embedding a Covert Channel in Active Network Connections," *Proc. IEEE GlobeCom,* 2009.

[24] S. Cabuk, C. Brodley, and C. Shields, "IP Covert Timing Channels: Design and Detection," *Proc. 11th ACM Conf. Computer and Comm. Security (CCS),* 2004.

[25] V. Berk, A. Giani, and G. Cybenko, "Detection of Covert Channel Encoding in Network Packet Delays," Technical Report TR2005536, Dept. of Computer Science, Dartmouth College, 2005.

[26] S. Gianvecchio and H. Wang, "Detecting Covert Timing Channels: An Entropy-Based Approach," *Proc. 14th ACM Conf. Computer and Comm. Security (CCS),* 2007.

[27] R. Walls, K. Kothari, and M. Wright, "Liquid: A Detection-Resistant Covert Timing Channel Based on IPD Shaping," *Computer Networks,* vol. 55, no. 6, pp. 1217-1228, Apr. 2011.

[28] S. Gianvecchio, H. Wang, D. Wijesekera, and S. Jajodia, "Model-Based Covert Timing Channels: Automated Modeling and Evasion," *Proc. 11th Int'l Symp. Recent Advances in Intrusion Detection (RAID),* 2008.

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

[30] Y. Liu, F. Armknecht, D. Ghosal, S. Katzenbeisser, A. Sadeghi, and S. Schulz, "Robust and Undetectable Covert Timing Channels for i.i.d. Traffic," *Proc. Information Hiding Conf.,* 2010.

[31] S. Zander, G. Armitage, and P. Branch, "Stealthier Inter-Packet Timing Covert Channels," *Proc. 10th Int'l IFIP TC 6 Conf. Networking,* 2011.

[32] X. Luo, E. Chan, and R. Chang, "TCP Covert Timing Channels: Design and Detection," *Proc. IEEE/IFIP Int'l Conf. Dependable Systems & Networks (DSN),* 2008.

[33] Y. Liu, D. Ghosal, F. Armknecht, A. Sadeghi, S. Schulz, and S. Katzenbeisser, "Hide and Seek in Time - Robust Covert Timing Channels," *Proc. 14th European Conf. Research in Computer Security (ESORICS),* 2009.

[34] W. Yu, X. Fu, S. Graham, D. Xuan, and W. Zhao, "DSSS-Based Flow Marking Technique for Invisible Traceback," *Proc. IEEE Symp. Security and Privacy,* 2007.

[35] Z. Ling, J. Luo, W. Yu, X. Fu, D. Xuan, and W. Jia, "A New Cell Counter Based Attack Against Tor," *Proc. 16th ACM Conf. Computer and Comm. Security (CCS),* 2009.

[36] X. Wang, S. Chen, and S. Jajodia, "Network Flow Watermarking Attack on Low-Latency Anonymous Communication Systems," *Proc. IEEE Symp. Security and Privacy,* 2007.

[37] A. Houmansadr, N. Kiyavash, and N. Borisov, "RAINBOW: A Robust and Invisible Non-Blind Watermark for Network Flows," *Proc. Network and Distributed Systems Security Symp. (NDSS),* 2009.

[38] A. Houmansadr and N. Borisov, "SWIRL: A Scalable Watermark to Detect Correlated Network Flows," *Proc. Network and Distributed Systems Security Symp. (NDSS),* 2011.

[39] X. Luo, P. Zhou, J. Zhang, R. Perdisci, W. Lee, and R. Chang, "Exposing Invisible Timing-Based Traffic Watermarks with BACKLIT," *Proc. 27th Ann. Computer Security Applications Conf. (ACSAC),* 2011.

[40] X. Luo, J. Zhang, R. Perdisci, and W. Lee, "On the Secrecy of Spread-Spectrum Flow Watermarks," *Proc. 15th European Conf. Research in Computer Security (ESORICS),* 2010.

[41] W. Hu, "Reducing Timing Channels with Fuzzy Time," *J. Computer Security,* vol. 1, pp. 233-254, 1992.

[42] J. Giles and B. Hajek, "An Information-Theoretic and Game-Theoretic Study of Timing Channels," *IEEE Trans. Information Theory,* vol. 48, no. 9, pp. 2455-2477, Sept. 2002.

[43] M. Kang, I. Moskowitz, and S. Chincheck, "The Pump: A Decade of Covert Fun," *Proc. 21st Ann. Computer Security Applications Conf. (ACSAC),* 2005.

[44] I. Moskowitz and M. Kang, "Covert Channels - Here to Stay?" *Proc. Ninth Ann. Conf. Computer Assurance Safety, Reliability, Fault Tolerance, Concurrency and Real-Time, Security and Security (COMPASS),* 1994.

[45] N. Ogurtsov, H. Orman, R. Schroeppel, and S. O'Malley, "Covert Channel Elimination Protocols," Technical Report TR96-14, The Univ. of Arizona, 1996.

[46] R. Yeung, *A First Course in Information Theory.* Kluwer Academic, 2002.

[47] H. Lee, E. Chang, and M. Chan, "Pervasive Random Beacon in the Internet for Covert Coordination," *Proc. Information Hiding Workshop,* 2005.

[48] D. Kreher and D. Stinson, *Combinatorial Algorithms: Generation, Enumeration and Search.* CRC press, 1998.

[49] H. Wilf, "East Side, West Side: An Introduction to Combinatorial Families with Maple Programming," http://www.cis.upenn.edu/~wilf/lecnotes.html, 2002.

[50] W. Myrvold and F. Ruskey, "Ranking and Unranking Permutations in Linear Time," *Information Processing Letters,* vol. 79, pp. 281-284, 2001.

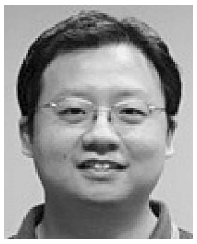
[51] B. Sharma and R. Khanna, "On m-ary Gray codes," *Information Sciences,* vol. 15, no. 1, pp. 31-43, 1978.

[52] X. Luo, E. Chan, P. Zhou, and R. Chang, "Supplemental Material to 'Robust Network Covert Communication Based on TCP and Enumerative Combinatorics'," 2012.

[53] X. Luo, E. Chan, and R. Chang, "CLACK: A Network Covert Channel Based on Partial Acknowledgment Encoding," *Proc. IEEE GLOBECOM,* 2009.

[54] "Packet Traces from WIDE Backbone,"http://tracer.csl.sony.co.jp/mawi/, 2012.

[55] V. Paxson, "LBNL/ICSI Enterprise Tracing Project," http://www.icir.org/enterprise-tracing/Overview.html, 2005.



Xiapu Luo received the PhD degree in computer science from the Hong Kong Polytechnic University in 2007 and then spent two years at the Georgia Institute of Technology as a post-doctoral research fellow. He is now a research fellow in the Department of Computing at the Hong Kong Polytechnic University. His current research focuses on network security and privacy, Internet measurement, and smartphone security. He is a member of the IEEE.

Edmond W.W. Chan received the BSc degree in information technology and the PhD degree in computer science from the Hong Kong Polytechnic University in 2005 and 2010, respectively. He is currently a researcher with Huawei Noah's Ark Laboratory, Hong Kong. His research interests include measurement and analysis of network traffic and networked systems.

Peng Zhou received the BSc and MSc degrees in Electrical Engineering from College of Donghua University in 2004 and 2007, respectively. He then joined Hewlett-Packard Company, Shanghai, China and worked as a firmware engineer and security software engineer for three years. He is now working toward the PhD degree in the Department of Computing at the Hong Kong Polytechnic University. His research interests cover network security and privacy. He is a student member of the IEEE.

Rocky K.C. Chang received the PhD degree in computer engineering from Rensselaer Polytechnic Institute. Immediately after that, he joined the IBM Thomas J. Watson Research Center working on performance analysis and simulation tools. He then joined the Department of Computing at the Hong Kong Polytechnic University, where he is now an associate professor. He is leading an Internet Infrastructure and Security Group, addressing problems in network security, network measurement, and network operations and management. He is a member of the IEEE.