# Novel Packet Size-Based Covert Channel Attacks against Anonymizer

Zhen Ling, *Member*, *IEEE*, Xinwen Fu, *Member*, *IEEE*,
Weijia Jia, Wei Yu, Dong Xuan, *Member*, *IEEE*, and Junzhou Luo, *Member*, *IEEE*

**Abstract**—In this paper, we present a study on the anonymity of Anonymizer, a well-known commercial anonymous communication system. We discovered the architecture of Anonymizer and found that the size of web packets in the Anonymizer network can be very dynamic at the client. Motivated by this finding, we investigated a class of novel packet size-based covert channel attacks against Anonymizer. The attacker between a website and the Anonymizer server can manipulate the web packet size and embed secret signal symbols into the target traffic. An accomplice at the user side can sniff the traffic and recognize the secret signal. In this way, the anonymity provided by Anonymizer is compromised. We developed intelligent and robust algorithms to cope with the packet size distortion incurred by Anonymizer and Internet. We developed techniques to make the attack harder to detect: 1) We pick up right packets of web objects to manipulate to preserve the regularity of the TCP packet size dynamics, which can be measured by the *Hurst* parameter; 2) We adopt the Monte Carlo sampling technique to preserve the distribution of the web packet size despite manipulation. We have implemented the attack over Anonymizer and conducted extensive analytical and experimental evaluations. It is observed that the attack is highly efficient and requires only tens of packets to compromise the anonymous web surfing via Anonymizer. The experimental results are consistent with our theoretical analysis.

**Index Terms**—Anonymizer, watermark, TCP dynamics

## 1 Introduction

In this paper, we explore a commercial anonymous communication system, *Anonymizer*, and present a novel class of packet size-based covert channel attacks that may drastically degrade the Anonymizer service. Traffic analysis is a common means to degrade communication privacy [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]. This covert channel exploits the varying size of packets through Anonymizer and is one type of active traffic analysis and has received much attention for its accuracy and efficiency recently [6], [7], [8], [10]. The idea of this technique is to actively embed a secret signal into the target traffic at one end of a communication channel and its accomplice recognizes the signal at the other end. Such attack can reduce the false positive rate significantly if the signal is long enough and does not require massive training of traffic for cross correlation analysis required in passive traffic analysis attacks [2], [3].

- *Z. Ling and J. Luo are with the School of Computer Science and Engineering, Southeast University, Nanjing 211189, P.R. China. E-mail: {zhenling, jluo}@seu.edu.cn.*
- *X. Fu is with the Department of Computer Science, University of Massachusetts Lowell, Lowell, MA 01854. E-mail: xinwenfu@cs.uml.edu.*
- *W. Jia is with the Department of Computer Science, City University of Hong Kong, Tat Chee Avenue, Kowloon, Hong Kong. E-mail: wei.jia@cityu.edu.hk.*
- *W. Yu is with the Department of Computer and Information Sciences, Towson University, Towson, MD 21252. E-mail: wyu@towson.edu.*
- *D. Xuan is with the Department of Computer Science and Engineering, The Ohio State University, Columbus, OH 43210. E-mail: xuan@cse.ohio-state.edu.*

*Manuscript received 29 May 2011; revised 16 June 2012; accepted 22 June 2012; published online 9 July 2012.*
*Recommended for acceptance by J. Wu.*
*For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-2011-05-0355.*
*Digital Object Identifier no. 10.1109/TC.2012.169.*

We will present the first exposure of the Anonymizer architecture, which consists of both anonymizing servers and client software. The Anonymizer server consists of reverse proxy/NAT, SSH server, and HTTP proxy, while the Anonymizer client software is a SSH port forwarding configuration tool. Using the Anonymizer client, we surfed various websites, including CNN, Yahoo, YouTube, and others and captured a large number of HTTP packets decrypted by the Anonymizer client. We found that the size of HTTP packets in Anonymizer network is very dynamic and random at the client, i.e., the number of non-MTU packets are much larger than that of MTU sized packets in the decrypted HTTP traffic. Motivated by this finding, we design a class of novel packet size-based covert channel attacks against the commercial Anonymizer service. In these attacks, the attacker between the malicious website and the victim client can embed a secret message into the packet size variation of target traffic. This attacker can be the owner of the malicious web server or one manipulating (repacketizing) the traffic between the web server and Anonymizer server. Without loss of generality, we use the former case as the example in this paper. An accomplice at the client side can sniff the traffic and recognize the secret message. Given the small size of the Anonymizer network, such sniffing is feasible to organizations or people with modest power. In this way, the anonymity provided by Anonymizer is compromised.

To make the attack harder to be detected by anyone other than attackers, we design various covert channel attacks to preserve the statistics of clean traffic (without message embedded) for modulated target traffic. Our baseline attack is to use the *Monte Carlo* sampling technique and sample the empirical cumulative distribution function (ECDF) of the clean web packet size. Then, random numbers embodying signals are carefully mapped into a sequence of packet sizes. In this way, the distribution of the modulated packet

size will be the same as the distribution of the clean traffic. However, the random sampling of this baseline attack may disturb the regularity and self-similarity of the TCP packet size dynamics [15], [16]. Measuring the *Hurst* parameter of the web traffic packet size sequence may expose the fact of the covert channel attack.

To overcome limitations of the baseline covert channel attack, we then introduce an enhanced covert channel attack, which can be made hard to detect and is described below: 1) To attack a HTTP session, we repacketize the web traffic into virtual web objects, and modulate secret messages bits into the size of last packets of these virtual web objects. The last packet of a web object is denoted as the least significant packet for brevity and clarity. The size of a least significant packet is very dynamic in comparison with other packet sizes. Modulation of successive packets to carry message bits will disrupt TCP packet size dynamics (as illustrated in Fig. 16), which can be measured by Hurst parameter from R/S plot and variance plot [15], [16]. This least significant packet-based covert channel approach can preserve TCP regularity and self-similarity (as illustrated in Fig. 15) while the attacker can control the number of virtual objects to control the number of message bits. 2) To preserve the size distribution of web packets of virtual web objects, we apply the *Monte Carlo* sampling technique to carefully sample the empirical cumulative distribution function of the least significant packet size of real web objects. This requires the input of the *Monte Carlo* method should be random and uniformly distributed. To this end, we first encrypt the message. The generated ciphertext bits are uniformly distributed and encoded into $k$-ary symbols. A $k$-ary symbol can then be mapped to a packet size by a *Monte Carlo* sampling technique. To cope with the packet size distortion caused by Anonymzier and Internet traffic dynamics (e.g., packet merging, limited TCP buffer, and various MTU), we design intelligent and robust detection algorithms to recover the message.

We have implemented these novel covert channel attacks against Anonymizer and performed extensive theoretical analysis and real-world experiments. The enhanced covert channel attack achieves high detection rate with *very low false positive rate*. The experimental results are consistent with our theoretical analysis. Notice that all our experimental attacks were conducted against the commercial Anonymizer service.

To the best of our knowledge, the attack presented here is the first exploiting the Anonymizer architecture and degrading its anonymity via packet size-based covert channel. The enhanced covert channel attack in the paper is simple, efficient, and hard to detect. Compared with related attacks [6], our attack requires just tens of packets to achieve high detection rate and low false positive rate. The attack is hard to detect for multiple reasons: 1) A successful attack session can be very short; 2) The packet size of traffic through the Anonymizer network is highly dynamic; The packet size observed at the client shows a large percentage of non-MTU packet size because of the Anonymizer client software's buffering mechanism and manipulation and the Internet traffic dynamics. It will be hard for the client to detect the attack; 3) The attack preserves the statistical properties of legitimate network traffic well. The client cannot detect the attack via measuring traffic dynamics. Considering privacy as a dual problem of digital forensics, our proposed techniques can be used by law enforcement to track malicious and anonymous web users via Anonymizer.

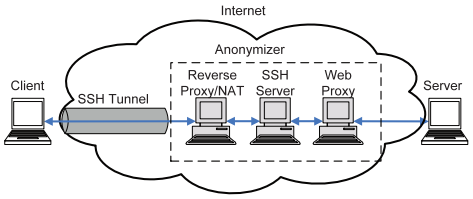


Fig. 1. Architecture of Anonymizer network.

The remainder of this paper is organized as follows: We explore the components of both Anonymizer server and client, and report our finding that size of HTTP packets in Anonymizer network is very dynamic in Section 2. In Section 3, we present a baseline covert channel attack scheme against Anonymizer based on Monte Carlo sampling and identify its limitations. In Section 4, we propose the enhanced covert channel attack scheme based on least significant packets. The attack is simple, accurate, efficient, and hard to detect. In Section 5, we analyze the detection rate and false positive rate of the attack. Extensive experimental results are presented in Section 6. We review related work in Section 7 and conclude this paper in Section 8.

## 2 Exploration of Anonymizer

In this section, we first present the Anonymizer architecture discovered by our passive inspection of traffic into and out of Anonymizer. We have replicated the discovered Anonymizer architecture in a lab environment and utilized the Anonymizer client software to browse the web through the lab Anonymizer servers. This verifies our discovery. We then show that the size of web packets in the Anonymizer client is very dynamic.

### 2.1 Architecture

The *Total Net Shield service* (*TNS*) from Anonymizer [17] is a commercially available anonymizing service. It is claimed that *TNS* (used with Anonymizer alternatively later for brevity and clarity) protects personal information by hiding the source computer's identity. Fig. 1 shows three basic components in such service: 1) *Anonymizer Client*: The client runs the commercial software to anonymize the client data. 2) *Anonymizer Server*: It consists of a reverse proxy/Network Address Translation (NAT) server, several SSH (Secure Shell) port forwarding servers, and proxy servers. 3) *Application Server*: It runs TCP applications such as web service:

1. *Components of Anonymizer Server*. The Anonymizer server consists of three components: a reverse proxy/NAT server, several SSH servers, and web proxies. The reverse proxy/NAT server dispatches inbound client traffic to SSH servers. For load balancing, reverse proxy/NAT uses a cluster of SSH servers and web proxy servers. For content privacy and communication anonymity, the client TCP traffic of POP3, SMTP, FTP, and HTTP is encrypted and sent to port 22 of a SSH server via SSH tunnel as shown in Fig. 1. The traffic is then decrypted and forwarded to port 80 of a web proxy. At last, the proxy server forwards the traffic to the destination. The reverse traffic follows the same path in the reverse order.

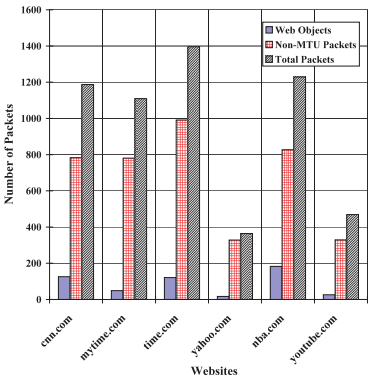


Fig. 2. Number of web objects, non-MTU packets, and total packets in the HTTP traffic observed at client.

The domain name of *Anonymizer* server is "cyberpass.net," which is resolved to a range of IP addresses, 168.143.113.110-120. We also observed that Anonymizer traffic to our client contains alternative IP addresses across sessions. Hence, we infer that Anonymizer implements the round robin Domain Name System (DNS) lookup mechanism for load balancing within its network. When a client looks up the domain name, Anonymizer DNS server resolves the domain names in a round robin fashion. Subsequent requests from the same client are then forwarded to the same Anonymizer server.

2. *Components of Anonymizer Client*. Client establishes a SSH connection to the SSH port forwarding server. The default cipher is AES-CBC with a 256-bit key (*AES256-CBC*). The default MAC is *HMAC-SHA1*. The client software sets the default domain name of Anonymizer SSH server and port as *cyberpass.net* and 22, respectively. The default local listening interface and proxy servers are 127.0.0.1:80 and *cyberpass.net:80*. For surfing on the web (the focus of this paper), the browser should use the local Anonymizer proxy so that the client may visit websites anonymously via Anonymizer servers. Users can manually configure the SSH server, local listening port, proxy server, encryption algorithm, and MAC algorithm in client software.

### 2.2 Dynamic Packet Size of HTTP Traffic in Anonymizer

For HTTP traffic, it may be common to see a sequence of MTU-sized packets followed by a shorter one as the last packet to transmit a web object in an extreme condition. However, our extensive experiments in the Anonymizer network show this claim does not hold in practice. Using the Anonymizer client, we surfed a number of websites, including CNN, Yahoo, YouTube, and others and captured a number of web packets out of SSH tunnel at the Anonymizer client. In Fig. 2, we show the number of web objects, non-MTU Packets and total Packets in the HTTP traffic, respectively. We can see that the number of non-MTU packet with random size is much larger than the number of web objects. We also found that in a sequence of packets transmitting one web object, a number of non-MTU packets are randomly located in different places.

These observations can be reasoned as follows: 1) The Anonymizer server, i.e., SSH server, repacks web packets. Note that the normal packet size in the unencrypted HTTP objects transmitted between HTTP server and Anonymizer server is the MTU size. However, Anonymizer server (i.e., SSH server) will add a SSH header and such padding increases the packet size. Hence, the repacked HTTP packet will be larger than the MTU size. These packets will then be split and the split packets will be merged with other SSH packets. These split packets will be repacked at the SSH client at the receiver side and this results in the non-MTU packets. 2) Network dynamics and performance of Anonymizer may also cause those non-MTU packets. If the network between HTTP server and Anonymizer server is congested, HTTP server will not have enough TCP windows to send out a MTU packet. If Anonymizer server is not overloaded at that moment, such packets will be delivered promptly. When this occurs, non-MTU packets will be generated.

In summary, the decrypted packet size observed at the client shows a large percentage of non-MTU packet size because of Anonymizer client software's packet manipulation and Internet traffic dynamics. It will be hard for the client to detect the enhanced attack in Section 4 by investigating whether non-MTU packets appear within the transmission of a web object. It is a nontrivial issue and we leave this as future work.

## 3 A Baseline Covert Channel Attack Based on Monte Carlo Sampling

Based on the finding that the packet size of HTTP traffic at the client in the Anonymizer network is very dynamic shown in Section 2.2, we consider to use the packet size variation of target traffic to embed a message. In this section, we first introduce the basic idea of a baseline covert channel attack based on the *Monte Carlo* sampling technique. We then give the details of the attack and discuss its limitations.

In attacks described from now on, the attacker between a malicious website and Anonymizer server can embed a secret message into the packet size variation of target traffic. This attacker can be the owner of the malicious web server or one manipulating (repacketizing) the traffic between the web server and Anonymizer server. Without loss of generality, we use theformer case to introduce the attacks in this paper.

### 3.1 Basic Idea

We assume that the attacker controls the malicious website and a reverse proxy that forwards the web traffic from the website. A client accesses the malicious website via Anonymizer. The attacker hosting the malicious website and his accomplice who is sniffing at the client side, try to confirm that this particular user is browsing the malicious website. The assumption is similar to those in [6].

The basic idea of this attack is as follows: An attacker at the malicious website controls the reverse proxy to embed a secret message into the web traffic. An accomplice of the attacker sniffs the traffic at the client side and determines if that client has received the traffic embedded with the same message. The message is embedded into the packet size variation of traffic. To be specific, a client accesses the malicious webpages via Anonymizer. The malicious website receives the HTTP requests for web objects via Anonymizer. It transmits the requested web objects to reverse proxy. The attacker at the reverse proxy generates a sequence of

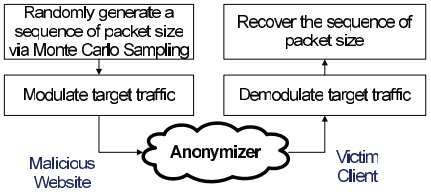


Fig. 3. Covert channel attack based on Monte Carlo sampling technique.

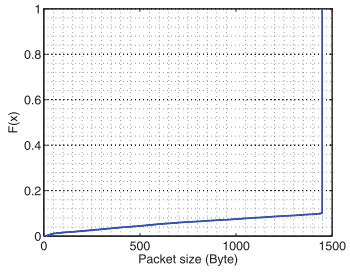


Fig. 4. Empirical CDF of packets size (Byte) between client and server.

random numbers between 0 and 1. By using the Monte Carlo sampling technique and the empirical cumulative distribution function of web packet size, the random numbers are carefully mapped into a sequence of packet sizes. In this way, the attacker ensures that the distribution of modulated packet size will not show abnormal and the message is difficult to be detected by others. The attacker modulates the size of web packets accordingly and forwards the packets to Anonymizer, which routes these modulated packets to the client. An accomplice at the client side sniffs the packets and derives the sequence of packet sizes. Although the packet sizes may be modified by Anonymizer, the accomplice can still recognize the modulated packet sizes via demodulation algorithm. If the sequence of derived packet sizes matches with the original one, the attackers can confirm that the client is browsing the malicious website. In this way, the anonymity of user is compromised.

Fig. 3 illustrates the workflow of the attack, which will be explained in details below. Note that in the following description, the demodulation of the target traffic and recovery of the sequence of packet size will be discussed in parallel with the generation of the sequence of packet size and modulation of the target traffic.

### 3.2 Generate and Recover Packet Size Sequence

*Generating a sequence of packet sizes.* Fig. 4[1] gives the empirical cumulative distribution function of the size of raw HTTP packets between the client and server from 30 well-known websites, including CNN, Yahoo, YouTube, and others. The raw HTTP packet size excludes the IP header and TCP header. We ignore the ACK packet because its raw packet size is zero. Let us denote the probability mass function (PMF) of the packet size as $\{p_1, p_2, \ldots, p_n\}$ corresponding to the sequence of packet sizes $\{pc_1, pc_2, \ldots, pc_n\}$, where $p_i$ is defined as the probability of packet size $pc_i$. The ECDF of raw HTTP packet size is below:

$$F_e(pc_i) = P(x \leq pc_i) = p_1 + p_2 + \cdots + p_i. \quad (1)$$

We use a random number generator to derive a sequence of random values $\{r_1, r_2, \ldots, r_l\}$, where $r \in [0, 1]$ and $l$ is the length of the sequence. To map a random number to a packet size, we select the first packet size $pc_i$ that meets the following condition:

$$F_e(pc_i) \geq r_i. \quad (2)$$

Fig. 5 illustrates this discrete Monte Carlo sampling process. Therefore, we derive the sequence of packet sizes, $\{p_{r0}, p_{r1}, \ldots, p_{rl}\}$. The mapping between the random number and the HTTP packet size can be written as

$$Table_1 : \{r_1, r_2, \ldots, r_l\} \Leftrightarrow \{p_{r0}, p_{r1}, \ldots, p_{rl}\}. \quad (3)$$

*Recovering the sequence of packet sizes.* The attacker at the reverse proxy modulates the packet size variation according to the sequence $\{p_{r0}, p_{r1}, \ldots, p_{rl}\}$. The web traffic with these modulated HTTP packet sizes will be forwarded to Anonymizer. Recall that the user establishes a SSH tunnel with Anonymizer and browses the website via the SSH tunnel. Because of the SSH padding mechanism and the fact that the HTTP proxy of Anonymizer filters HTTP header fields, the raw HTTP packet sizes are changed. To accurately recognize the embedded symbols, the sniffer at the client side records the SSH packets, and obtains the SSH packet size based on raw HTTP packets.

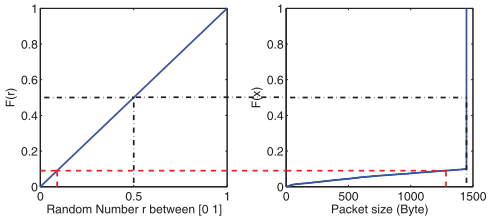


Fig. 5. Mapping a random number to a packet size based on Monte Carlo sampling technique.

### 3.3 Modulate and Demodulate Target Traffic

From Section 2, we know that a SSH tunnel is established between a client and Anonymizer to protect sender anonymity. The web traffic is encrypted by SSH, which consists of three major components: 1) the transport layer protocol [18] provides server authentication, confidentiality, and integrity, 2) the authentication protocol [19] authenticates the client to the server, and 3) the connection protocol [20] multiplexes several logical channels into the encrypted tunnel. The attacker sniffing at the client side can ignore the SSH packets generated by authentication protocol and target only SSH packets generated by connection protocol.

Fig. 6 illustrates the structure of a SSH packet. The packet consists of an IP header, a TCP header, and the SSH packet [18]. The SSH packet consists of ciphertext and MAC. The ciphertext consists of a length field of 4 bytes, a padding length field of 1 byte, a data payload field of $X$ bytes, and a random padding of $Y$ bytes. A minimum of four padding bytes must be added. The padding should be random and should ensure that the length of the SSH package must be multiple of cipher block size or 8, whichever is larger. The maximum length of padding is 255 bytes. In our case, the encryption algorithm is *AES256-CBC* by default. The AES cipher has a block size of 128 bits and key size of 256 bits. Therefore, the total number of bytes in the ciphertext should be a multiple of the cipher block size of 16 bytes, i.e., $(4 + 1 + X + Y)\ mod\ 16 = 0$. The MAC algorithm is *HMAC-SHA1* by default and the length of the MAC value is 20 bytes.

1. Fig. 4 is observed by the attacker between the client and Anonymizer server. It is different from Fig. 2, which is observed by the client for decrypted traffic.

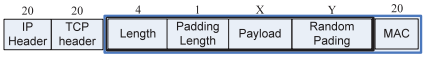


Fig. 6. SSH packet format.


Fig. 7. Channel message format.

Once the SSH tunnel is established, the client and server may open single or multiple channels. Note that these multiple channels are multiplexed into a single SSH tunnel. The data are packed into the channel message as the payload of SSH packet. Specifically, the channel message uses the packet whose "Message Type" field is SSH_MSG_CHANNEL_DATA. The SSH_MSG_CHANNEL_DATA packet contains the HTTP data. Fig. 7 illustrates the structure of channel message. It includes a message type field of 1 byte, a recipient channel field of 4 bytes, a data length field of 4 bytes, and a data field of $Z$ bytes. The "message type" defines the type of message. The "recipient channel" is channel number given in the original open request. The "data length" indicates the size of the data.

Considering above facts, we propose an algorithm to derive SSH packet size based on raw HTTP packet. Note that in our case, attacker only cares about SSH_MSG_CHANNEL_DATA packets. The SSH packet size is discrete because of SSH padding mechanism. We assume that the MTU packet is 1,500 bytes. Fig. 8 illustrates all possible lengths of SSH packet between the client and Anonymizer server.

In terms of the sequence of raw HTTP packet sizes, $\{p_{r0}, p_{r1}, \ldots, p_{rl}\}$, the attacker can deduce a corresponding sequence of SSH packet sizes $\{ps_{r0}, ps_{r1}, \ldots, ps_{rl}\}$, excluding the IP header and the TCP header by using Algorithm 1. Hence, (3) can be converted into

$$Table_1' : \{p_{r0}, p_{r1}, \ldots, p_{rl}\} \Leftrightarrow \{ps_{r0}, ps_{r1}, \ldots, ps_{rl}\}. \quad (4)$$

Once such a sequence of SSH packet sizes is recognized at the client side, the communication relationship between the malicious website and the client is confirmed. Hence, the anonymity of user is compromised.

**Algorithm 1.** Calculating the SSH packet size based on raw HTTP packet.

**Require**:

(a) $S$, the size of the raw HTTP packet, (b) $S_h$, the size of the SSH header, 5 bytes, (c) $S_c$, the size of the SSH channel header, 9 bytes, (d) $S_p$, the size of the padding, (e) $S_b$, the cipher block size, 16 bytes, (f) $S_a$, the size of MAC, 20 bytes, (g) $S_s$, the size of the SSH packet.

**Ensure**: return $S_s$

1: $S_p = S_b - (S_h + S_c + S)\ mod\ S_b$
2: **if** $S_p < 4$ **then**
3: $\quad S_p = S_b + S_p$
4: **end if**
5: $S_s = S_h + S_c + S + S_p + S_a$

### 3.4 Limitations of the Attack

From the description of the attack, we can see that to maintain the original distribution of web packet size, the attacker randomly samples the distribution of web packet sizes and generates a packet size sequence used as signal. However, this process may disturb the regularity of TCP packet size dynamics [15], [16]. In the following, we study the impact of the baseline attack on TCP dynamics. We first define the self-similarity related to TCP traffic and then introduce two methods to measure this property. Our real-world evaluation data in Section 6 further demonstrate that this attack as well as related attacks in [21], [22] becomes detectable to targeted users because of its disruption of TCP dynamics. The following discussion of self-similarity and R/S-Statistics is credited to related work [15], [16] while we innovatively apply them in our context.

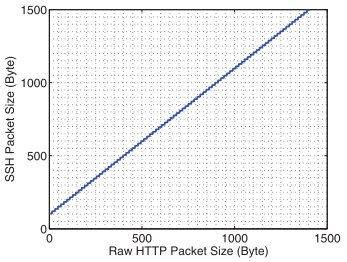


Fig. 8. Raw HTTP packet size versus SSH packet size enveloped by IP header and TCP header.

#### 3.4.1 Self-Similarity

Consider a random series $X = (X_t : t = 0, 1, \ldots, n-1)$. It has mean $\mu = E[X_t]$, variance $\sigma^2 = E[(X_t - \mu)^2]$, and autocorrelation function

$$r(k) = \frac{E[(X_t - \mu)(X_{t+k} - \mu)]}{E[(X_t - \mu)^2]},$$

where $k = 0, 1, \ldots, n-1$.

For each $m \geq 1$, let $X^{(m)} = \{X_k^{(m)} : k = 1, 2, \ldots, n-1\}$ be a new series obtained by averaging the original series $X$ over nonoverlapping blocks of size $m$, where

$$X_k^{(m)} = \frac{1}{m} \sum_{i=(k-1)m+1}^{km} X(i), k \geq 1. \quad (5)$$

Let $r^{(m)}$ be the corresponding autocorrelation function of the aggregated series $X^{(m)}$.

The process $X$ is self-similar if the corresponding aggregated processes $X^{(m)}$ have the same correlation structures as $X$, i.e., $r^{(m)}(k) = r(k)(k = 1, 2, \ldots)$, for all $m = 1, 2, \ldots, n-1$. Specifically, when the aggregated process $X^{(m)}$ is the same as $X$, $X$ is exactly self-similar. If $r^{(m)}$ agrees asymptotically with $r(k)$ of $X$, i.e., $r^{(m)}(k) \to r(k)$, as $m \to \infty$ $(k = 1, 2, \ldots)$, the process $X$ is called (asymptotically) second-order self-similar. In other words, the process $X$ is asymptotically second-order self-similar if the aggregated process $X^{(m)}$ is indistinguishable from the process $X$. If $r^{(m)}$ agrees asymptotically with zero, i.e., $r^{(m)} \to 0$ $(k = 1, 2, \ldots)$, as $m \to 0$, the process $X$ is a second-order pure noise.

The self-similar process defined above comes from the explanation and interpretation of one empirical law commonly referred to as the *Hurst*'s law or the *Hurst* effect [23]. Give a set of observations $(X_k : k = 1, 2, \ldots, n)$ with sample mean

$$\bar{X}(n) = \frac{1}{n} \sum_{t=1}^{n} X(t), \quad (6)$$

and sample variance

$$S^2(n) = \frac{1}{n}\sum_{t=1}^{n}(X_t - \bar{X}(n))^2. \quad (7)$$

Then, R/S statistics (or the rescaled range) is given by

$$\frac{R(n)}{S(n)} = \frac{1}{S(n)}\left(\max_{1\le k\le n}\left\{\sum_{t=1}^{k}(X_t - \bar{X}(n))\right\} - \min_{1\le k\le n}\left\{\sum_{t=1}^{k}(X_t - \bar{X}(n))\right\}\right). \quad (8)$$

If the series of observations have the long-range dependence, it follows the following relation [15], [24]:

$$E\left[\frac{R(n)}{S(n)}\right] \sim \alpha n^H, as\ n \to \infty, \quad (9)$$

where *Hurst* parameter $H > 0.5$, and $\alpha$ is a finite positive constant. On the other hand, Mandelbrot and Ness in [25] found that if the relation follows:

$$E\left[\frac{R(n)}{S(n)}\right] \sim \gamma n^{0.5}, as\ n \to \infty, \quad (10)$$

where *Hurst* parameter $H = 0.5$, and $\gamma$ is a finite positive constant, the underlying process is purely random.

To estimate the degree of self-similarity $H$ (*Hurst* parameter), we can use two different methods: 1) analysis of the $R/S$-statistics, and 2) analysis of the variance of the aggregated process $X^m$, which will be discussed below.

#### 3.4.2 $R/S$-Statistics

The $R/S$-statistics relies on the fact that for a self-similar data set, the rescaled range grows according to a power law with the exponent parameter $H$. The objective of analyzing R/S statistics of an empirical data set is to infer the degree of self-similarity $H$ in (9) for a self-similar process. Given a sequence of $N$ observations $X_k : k = 1, 2, \ldots, N$, all samples can be divided into $K$ nonoverlapping blocks, where $K = \lfloor \frac{N}{n} \rfloor$. Hence, we can calculate the rescaled range $\frac{R(t_i,n)}{S(t_i,n)}$ defined in (8), where $t_i = \frac{(i-1)\cdot N}{K} + 1 (i = 1, 2, \ldots, N)$ and $(t_i - 1) + n \le N$. $S^2(t_i, n)$ are sample variances of $X_{t_i}$, $X_{t_i+1}, \ldots, X_{t_i+(n-1)}$ defined in (7). Then, we can plot $\log(\frac{R(t_i,n)}{S(t_i,n)})$ versus $\log(n)$ and obtain, for each $n$, several points on the plot called the rescaled range plot (or the pox diagram for $R/S$ statistics). *Hurst* parameter $H$ defined in (9) and (10) can be estimated by fitting a line of a certain slope to accommodate all points in the pox plot. Then, the slope of the line is an estimate of $H$.

For a long-range dependent process, points in the $R/S$ plot are scattered around a straight line with slope greater than $\frac{1}{2}$, i.e., $H > \frac{1}{2}$, for sufficiently large $n$. If the $R/S$ statistics is scatted around a straight line with a slope of around 0.5, i.e., $H \approx 0.5$, it indicates that the data are pure white noise.

#### 3.4.3 Variance of the Aggregated Process

Other statistics can also be used to measure *Hurst* parameter $H$. The most important feature of self-similarity is that the variance of the sample mean decreases more slowly than the reciprocal of the sample size (i.e., slowly decaying variance). That is, $Var(X^{(m)}) \sim \eta m^{-\beta}, as\ m \to \infty$, where $\eta$ is a positive constant. If the aggregated series $X^{(m)}$ is second-order pure noise, we have $Var(X^{(m)}) \sim \theta m^{-1}, as\ m \to \infty$, where $\theta$ is a positive constant and $X^{(m)}$ is given by (5). In the variance plot, the variance of the aggregated process $X^{(m)}$ is plotted against $m$ on a log-log scale. If the points in the variance plot are scattered randomly around a straight line with the slope between $-1$ and 0, i.e., $0 < \beta < 1$, it indicates that the aggregated process $X^{(m)}$ has long-rang dependence and is self-similarity. For pure random noise, the points in variance plot are scattered around a straight line with a slope around $-1$, i.e., $\beta \approx 1$. Notice that $H = 1 - \frac{\beta}{2}$, and the *Hurst* parameter $H$ can be estimated from $\beta$.

Therefore, it is trivial for a user to detect this baseline covert channel attack. The user can record the packet size of the web traffic, and use the packet size sequence to measure the Hurst parameter via R/S method and variance methods. This does not affect the original traffic or incur much overhead. We conducted the real-world experiments and Figs. 16 and 19 demonstrate that the Hurst value of the packet size sequence under this simple attack is around 0.5 and implies pure randomness. That is, the baseline covert channel attack can disrupt the self-similarity of a web packet size sequence. Consequently, it can be readily detected via the measurement of Hurst parameter.

## 4 An Enhanced Covert Channel Attack Based on Least Significant Packets

To overcome the limitations of the baseline covert channel attack in Section 3, we design an enhanced covert channel attack by carefully picking up the right packets of web objects to greatly preserve the regularity of the TCP packet size dynamics measured by the *Hurst* parameter. In particular, we virtually generate web objects with different sizes, repacketize the web traffic, and modulate the size of the last packet of each virtual web object. Analogous to the least significant bit used for information hiding in images, we denote the last packet of a web object as the least significant packet for brevity and clarity. In the following, we first introduce the least significant packets, and then present the detailed workflow. Lastly, we discuss some practical issues and present our solutions.

### 4.1 Least Significant Packets

As shown in Fig. 4, normal HTTP packets (not through Anonymizer) can be roughly categorized into two classes. The *Class I* packet is defined as the largest packet in the normal HTTP traffic, i.e., 1,500 bytes in case of Ethernet, including an IP header of 20 bytes and a TCP header of 32 bytes. The actual size of the HTTP content in the TCP payload is 1,448 bytes. The *Class II* packet has a size less than 1,500 bytes. Such packets are usually generated by the "tail" of the web object, i.e., the last packet when the web object is downloaded. If the web object size including the HTTP header is $w$ bytes, the size of the "tail" of that web object is $(w\ mod\ 1,448) + 20 + 32$ bytes. Here, we denote *Class II* packets as *least significant packets* for brevity and clarity. This is analogous to the least significant bit used for information hiding in images [26].

By analyzing the traffic from 30 well-known websites including CNN, Yahoo and YouTube, we obtain the ECDF of the size of raw HTTP packets in *Class II* as shown in Fig. 9. As shown in Fig. 4, the raw HTTP packet size does not include the IP header and the TCP header. Also, the ACK packet is ignored because of its zero length. The PMF of the

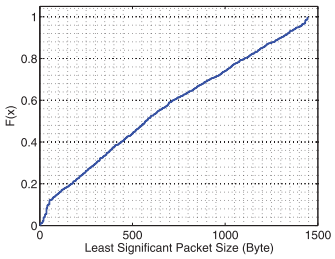


Fig. 9. Empirical CDF of the least significant packet size.

least significant packets is $\{p'_1, p'_2, \ldots, p'_m\}$ and the corresponding packet sizes are $\{pc'_1, pc'_2, \ldots, pc'_m\}$. $p'_i$ is the probability of the packet size $pc'_m$. Hence, the ECDF of the least significant packet size can be formalized as follows:

$$F_{lsp}(pc'_i) = P(x' \leq pc'_i) = p'_1 + p'_2 + \cdots + p'_i. \quad (11)$$

As per our observations from Fig. 4, the overall probability of the least significant packets observed by the attacker between the client and the Anonymizer server is no more than 10 percent. In addition, the size of such packets is more random and dynamic. Because of these, an attacker can use those packets to embed secret symbols into the web traffic and expect to preserve TCP regularity and self-similarity.

### 4.2 Enhanced Idea of Covert Channel Attacker Based on Least Significant Packets

An attacker at the malicious website controls the reverse proxy to embed a secret message into the web traffic. An accomplice of the attacker sniffs the traffic at the client side and determines if that client has received the traffic embedded with the secret message. The message can be represented as a sequence of symbols (for example, "0000" to "1111") and one symbol corresponds to one packet size. We virtually generate web objects with different sizes, repacketize the web traffic, and choose appropriate size of a least significant packet for a message symbol.

To make the covert channel hard to detect, we need to preserve the least significant packet size ECDF in Fig. 9. This is the criterion for mapping a symbol to a least significant packet size. A classical way for preserving an ECDF is *Monte Carlo* sampling. In our case, we divide the $y$-axis $[0, 1]$ of the ECDF into equal segments such as the 16 segments in Fig. 11. The 16 segments corresponding to symbols from "0000" ("0") to "1111" ("F"). Hence, one symbol can be uniformly mapped to a few packet sizes along the $x$-axis of the ECDF. We need to guarantee that one symbol corresponds to at least one packet size. To preserve the ECDF, Monte Carlo sampling requires the message has uniformly distributed symbols. To achieve this, we encrypt the message first and transmit the cyphertext over the covert channel. A strong cipher generates uniformly distributed symbols in the ciphertext [27].

### 4.3 Workflow

Fig. 10 shows the workflow of the attack based on least significant packets. An attacker at the malicious website (reverse proxy) selects a message and encrypts it. The attacker first selects a message signal and encrypts the message by a strong cipher. Note that a strong cipher eliminates any recognizable pattern in the message [27] and the ciphertext appears random. The cyphertext as the output

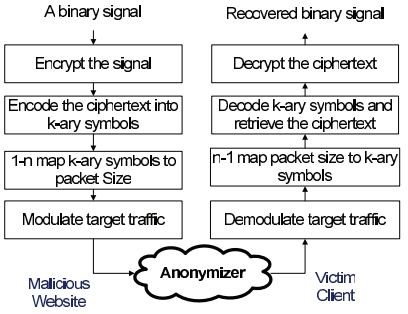


Fig. 10. Workflow of covert channel attack based on least significant packets.

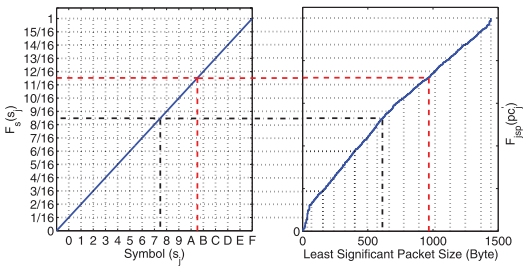


Fig. 11. Mapping between symbols and least significant packet sizes.

of encryption consists of a sequence of binary bits. The attacker then encodes the cyphertext into $k$-ary symbols. For example, 4 bits corresponds to a hexadecimal symbol. To avoid disturbing the distribution of the packet size, the attacker carefully maps the symbols to packet sizes based on the distribution of empirical least signal packet size. Therefore, the least significant packets work as a covert channel and carry information.

The attacker chooses appropriate time and modulates the size of least significant packets to embed the secret symbols into the target traffic while preserving its TCP regularity and self-similarity property. The traffic embedded with symbols goes through Anonymizer and arrives at the client side. Because of network dynamics and processing capacity of Anonymizer, the packets carrying embedded symbols may be combined with the fore-and-aft packets at Anonymizer and the symbols can be distorted. We develop an intelligent and robust detection algorithm to recover symbols from distorted packet sizes. Hence, the attacker's accomplice at the client side can sniff the target traffic out of Anonymizer and recover the embedded symbols. The attacker can decrypt the encrypted message. If the decrypted message is same as the original one, the attacker confirms the communication relationship between the user and malicious website. Sender anonymity promised by Anonymizer is then compromised. The detailed workflow steps are given in the following sections.

### 4.4 Embedding a Signal at the Malicious Website

*Step 1: Encrypting the signal*. At the malicious website, the attacker selects a message signal and encrypts the message by a strong cipher. An attacker can use $RC4$ in counter mode with an initialization vector (IV) to encrypt the message and make the ciphertext appear random. Our experiments confirm that the RC4 cyphertext hexdecimals have a uniform distribution. Given a message of $u$ bytes, we derive the encrypted message as

$$E_{rc4}(Message, IV) \rightarrow \{B_1, B_2, \ldots, B_{8u}\}, \quad (12)$$

where $\{B_1, B_2, \ldots, B_{8u}\}$ is the sequence of the binary bits.

*Step 2: Encoding the cyphertext into k-ary symbols*. To guarantee that one symbol is mapped to at least one packet size, we need to carefully choose the total number ($k$) of symbols representing a message. The ECDF of the least significant packet size is derived in (11) and we can find the maximum $k$ such that

$$2^k \leq 1/p'_{max}, \quad (13)$$

where $p'_{max}$ is the maximum value in the PMF of the least significant packet size. Then, $\{0, 1, \ldots, 2^k - 1\}$ become our symbols, i.e., $k$-bits is used to represent one $k$-ary symbol. Using real-world data, we have $p'_{max} = 0.027$ and derive $k = 5$ based on (13). In our case, we choose $k = 4$, corresponding to 4-ary symbols. Let the symbols be $\{s_0, s_1, \ldots, s_{2^k-1}\}$. The sequence of the binary bits can be translated into

$$\{B_1, B_2, \ldots, B_{8u}\} \Leftrightarrow \{S_1, S_2, \ldots, S_{8u/k}\}, \quad (14)$$

where $8u/k$ is the length of the symbols (padding is needed if $8u \bmod k \neq 0$), $\{S_1, S_2, \ldots, S_{8u/k}\}$ are the sequence of the encrypted symbols, where $S_i$ is the $i$th symbol.

Step 3: *Mapping k-ary symbols to packet sizes.* We use the approach of Monte Carlo sampling in Fig. 11 to map a ciphertext symbol to an appropriate least significant packet size. Because of the random ciphertext, the probabilities of ciphertext symbols are equal, $1/2^k$. Then the CDF of the symbols is $F_s(s_j) = \frac{j}{2^k}$. To map a $k$-ary symbol $s_j$ to the corresponding packet size, we need to find the scope of packet sizes that can be mapped to the $k$-ary symbol. The upper bound $pc'_i$ of a scope for $s_j$ is the largest packet size satisfying the following inequality, $F_{lsp}(pc'_i) = p'_1 + p'_2 + \cdots + p'_i \leq F_s(s_j) = \frac{j}{2^k}$.

Let the corresponding packet sizes be $\{p_{s0}, p_{s1}, \ldots, p_{s(2^k-1)}\}$. We can make a mapping from symbol $s_j$ to one scope $(p_{s(j-1)}, p_{sj}]$, where $j \in \{0, 1, \ldots, 2^k - 1\}$. To maintain the distribution of the least significant packets, we adopt the Monte Carlo sampling approach to uniformly pick up a packet size $p'_{sj}$ from the scope $(p_{s(j-1)}, p_{sj}]$ for mapping the symbol $s_j$ as illustrated in Fig. 11 to preserve the least significant packet size ECDF. We have the following mapping table between symbols and packet sizes:

$$Table_2 : \{s_0, s_1, \ldots, s_{2^k-1}\} \Leftrightarrow \{p'_{s0}, p'_{s1}, \ldots, p'_{s(2^k-1)}\}. \quad (15)$$

By utilizing the above mapping, in our case, 16 different least significant packet sizes $\{p'_{s0}, p'_{s1}, \ldots, p'_{s15}\}$ are mapped to the hexadecimal symbols, $\{0, 1, \ldots, E, F\}$, one time, as illustrated in Fig. 11. Consequently, 1 byte in the ciphertext can be encoded into two packet sizes. Equation (15) can be converted into

$$Table'_2 : \{0, 1, \ldots, F\} \Leftrightarrow \{p'_{s0}, p'_{s1}, \ldots, p'_{s15}\}. \quad (16)$$

Thus, we can derive the sequence of the corresponding packet sizes $\{P_{S1}, P_{S2}, \ldots, P_{S(2u)}\}$ in (14).

Because of the fact that ciphertext generates uniformly distributed symbols and the $1-$n mapping from one symbol to packet size (Notice that one symbol can be uniquely represented by corresponding multiple packet sizes), the least significant packet size distribution is not changed via this modified Monte Carlo sampling approach.

*Step 4: Modulating target traffic*. When a client accesses the malicious website, the attacker starts to embed a message into the target web traffic. Because the IP addresses range of Anonymizer is small, the attacker can simply use those IP addresses to identify whether the incoming HTTP request is from Anonymizer. The attacker can identify the first arriving HTTP "GET" request[2] and derive object length from HTTP headers, and embed the secret symbols generated in the previous steps into the web traffic.

To avoid disturbing the TCP traffic dynamics and make it difficult for clients and third parties to detect the attack from traffic pattern, the attacker carefully chooses the appropriate symbol location in the flow and modulates the packet size to embed a message. According to the previous study in [28], [29], the web object size is best fitted by a Lognormal distribution. Considering this, we virtually generate objects of different size and calculate the location of the least significant packet, i.e., the number of packets between successive symbols. Denote the location of our symbols as $\{o_1, o_2, \ldots, o_{2u}\}$. The first symbol $o_1$ is located at the number of packets counted from the first HTTP response packet to the first symbol.

To improve attack performance, the attacker can modulate a preamble of couple of packets at the location $o_1$ before the symbol. When the browser fetches the main object, it may establish several persistent TCP connections and send successive HTTP requests for other linked objects. To establish corresponding new channels, the SSH client establishes the channels and sends the SSH channel messages to Anonymizer. The "Message Type" field of these channel messages is SSH_MSG_CHANNEL_OPEN. The Anonymizer SSH server opens the channels and responds with the SSH_MSG_CHANNEL_OPEN_CONFIRMATION messages to the client. Then, the SSH client forwards these HTTP requests to the browser. We use the SSH_MSG_CHANNEL_OPEN_CONFIRMATION message (52 bytes excluding the IP header and TCP header) and five, for example, such successive SSH packets as the symbol preamble. Therefore, each raw HTTP packet size is 11 bytes. With the channel header, SSH header, padding, and MAC, the packet reaches 52 bytes. Such a symbol preamble will not incur much attention from the client.

To reliably modulate packets to desired sizes, the attacker at the reverse proxy can modify the size of the proxy read buffer. The read buffer size is 1,448 bytes. The attacker counts the number of packets to Anonymizer and when number of the transmitted packets is equal to $o_i$, the attacker sets up the read buffer to $p_{si}$ and resets the counter. $p_{si}$ bytes of the next incoming packet is pushed into read buffer, flushed into write buffer, and then sent to Anonymizer. The rest of the packets will stay in TCP buffer and wait for the next read event.

### 4.5 Recovering the Signal at the Client Side

*Step 1: Demodulating target traffic*. An accomplice of the attacker at the client side sniffs the SSH traffic transmitted to the user and records the packet size. Recall the accomplice knows the sequence of the raw HTTP packet sizes $\{P_{S1}, P_{S2}, \ldots, P_{S(2u)}\}$ and the sequence of the packet interval $\{o_1, o_2, \ldots, o_{2u}\}$. We then calculate the sizes of SSH packets corresponding to raw HTTP packets. Denote the sequence

2. We use the GET request as an example. However, it is easy to apply this attack to other request methods.

of the SSH packet sizes as $\{P_{ssh1}, P_{ssh2}, \ldots, P_{ssh(2u)}\}$. Then, the attacker is able to find out the SSH packet size carrying the symbols.

*Step 2: Mapping packet sizes to k-ary symbols*. Using the same mechanism described in Algorithm 1, the accomplice can record the size of SSH packet $P_{sshi}$ carrying the symbols and obtain the corresponding raw HTTP packet size $P_{Si}$. The obtained HTTP packet sizes will be mapped to the symbols, i.e., the hex characters, using (16). The accomplice now derives the sequence of the mapped symbols $\{S_1, S_2, \ldots, S_{2u}\}$.

*Step 3: Decoding **k**-ary symbols into the signal*. Given the sequence of recovered symbols $\{S_1, S_2, \ldots, S_{2u}\}$ from the previous step, the attacker applies (14) to translate the hexdecimal symbols into binary bits $\{B_1, B_2, \ldots, B_{8u}\}$.

*Step 4: Decrypting the ciphertext*. Finally, the message is decrypted using RC4 algorithm with IV as

$$D_{rc4}(\{B_1, B_2, \ldots, B_{8u}\}, IV) \rightarrow Message. \tag{17}$$

If the decrypted message is the same as the original one, the attackers confirm the relationship between the user and malicious website.

### 4.6 Issues and Solution

Because of network dynamics and the processing capacity of Anonymizer, the packets carrying the embedded symbols may be combined with the fore-and-aft packets at Anonymizer and the symbols may be distorted. This may affect the effectiveness of the attack. To address this issue, we present an intelligent and robust detection algorithm.

#### 4.6.1 Scenarios of Interference

In the following, we discuss several scenarios of interference, which will distort the least significant packets carrying the symbols and reduce the symbol detection rate.

*Packet merging*. Given a sequence of packet arrival times from the reverse proxy to Anonymizer $\{T_1, T_2, \ldots, T_q\}$, where $q$ is the number of packets. Let $T_p$ be the processing time at Anonymizer, including the processing time of HTTP proxy, SSH server, queuing delay, and so on. We assume that the HTTP packet carrying the symbol arrives at $T_i$ and the successive normal HTTP packet arrives at $T_{i+1}$. The symbol packet and the successive normal packet will merge if

$$T_i + T_p \geq T_{i+1} (0 \leq i \leq u-1), \tag{18}$$

$$T_p \geq T_{i+1} - T_i. \tag{19}$$

Similarly, if

$$T_{i-1} + T_p \geq T_i (1 \leq i \leq u), \tag{20}$$

$$T_p \geq T_i - T_{i-1}, \tag{21}$$

the symbol packet will be distorted by merging with the previous packet. Specifically, if the SSH server cannot process the HTTP packets promptly, the following packet will be combined with the symbol packet in the SSH buffer, for example, in the case that the network is congested between the user and Anonymizer.

To overcome the issue of packet merging, the attacker at the reverse proxy can add a delay interval $I$ before reading the symbol packet and after writing the symbol packet out, i.e., $T_{i+1} = T_i + I + D$ and $T_{i-1} + I + D = T_i$, where $D$ is the delay between the reverse proxy and Anonymizer. Therefore, if we can appropriately select the delay interval, i.e., $T_p \geq I + D$, and the following conditions are met, the normal HTTP packets will not be combined with symbol packets:

$$T_p \geq T_{i+1} - T_i \qquad or \qquad T_p \geq T_i - T_{i-1}, \tag{22}$$

$$T_p \geq I + D. \tag{23}$$

*Shift of symbol location*. In some cases, the data pulled from the TCP buffer are smaller than the symbol packet size. For example, the symbol location may coincide with the location of a real least significant packet if it is smaller than the symbol packet size because web objects are fetched randomly. When this happens, the attacker can simply ignore this packet and modulate the next packet size. Of course, in this way the symbol location sequence $\{o_1, o_2, \ldots, o_{2u}\}$ would be changed. However, the accomplice can carefully select a toleration scope and search the correct packet. We will discuss the details of this solution in Section 4.6.2.

*MTU*. The MTU may also distort the packets carrying symbols. The normal raw HTTP packet size is 1,448 bytes. After raw HTTP packet is processed by SSH server at Anonymizer, the SSH packet size will reach 1,492 bytes. Then, the SSH packet size will be 1,544 bytes, including an IP header of 20 bytes and a TCP header of 32 bytes. Because of MTU, the SSH packet will be split into two parts, one has 1,500 bytes and the other has 44 bytes. The first 1,500 bytes will be packed into a packet and forwarded to the user and second 44 bytes will stay in TCP buffer. If the network condition is good between the user and Anonymizer SSH server, that 44 bytes will be added to an IP header and a TCP header, then it will be forwarded to the user in one separate packet. Otherwise, the 44 bytes will be combined with the next normal SSH packet or our symbol packet in TCP buffer. More complicated situations may occur. For example, two raw HTTP packets may be combined in SSH buffer. The merged packet will reach 2,948 bytes, and it is then split into three parts: two have 1,500 bytes and the rest has 104 bytes. There are many other cases of combinations.

*HTTP proxy filtering*. The Anonymizer HTTP proxy may filter the HTTP header field "Connection: close\r\n," which is 19 bytes including the blank behind the colon. The normal raw HTTP packet size of 1,448 bytes becomes 1,429 bytes. Then, the packet with 1,476 bytes is processed by the SSH server. By adding the IP header and TCP header, it will reach 1,528 bytes. Because of MTU, it will be split into 1,500 and 28 bytes. Therefore, the rest 28 bytes may be combined with later symbol packets.

#### 4.6.2 An Intelligent Detection Algorithm

To deal with the issues of limited TCP buffer and MTU, we design an intelligent detection algorithm (Algorithm 2). Because of the MTU, the packets through Anonymizer can generate a small packet of 44 bytes and shift the predefined symbol location $o_i$ (the number of the packets between the $i$th symbol packet and the last preamble packet). Denote the sequence of the practical SSH packet size as $Q$ and the starting point for search the $i$th practical symbol packet from $Q$ as $s_i$, where $s_1 = o_1$. To reduce false positive rate, we

introduce a threshold $S_t$, to search the practical location of the $i$th symbol packet $o'_i$ from $Q$ between $s_i$ and $s_i + S_t$. $o'_i$ is the number of the packets between the $i$th practical symbol packet and the last preamble packet. Once practical location $o'_i$ is found, we update $s_i = o'_i$ and continue to search the next symbol between $s_i$ and $s_i + S_t$.

**Algorithm 2.** Detection Algorithm.
**Require**:
(a) $Q$, the sequence of the total SSH packets sizes, (b) $L$, the length of the symbols, (c) $P_{ssh}$, the sequence of the SSH symbol packet sizes, (d) $i$, the index of $Pa_{ssh}$, (e) $o[i]$, the $i$th predefined symbol packet location, (f) $o'[i]$, the $i$th practical symbol packet location, (g) $S_t$, the threshold, (h) $s[i]$, the starting point of the $i$th symbol, (i) $m$, the number of the matched symbols.
**Ensure**: return m

```
s[1] = o[1]
for i = 1 to L do
    if P_ssh[i] == (Q between [s[i], s[i] + S_t]) then
        Record o'[i]; s[i + 1] = o'[i];
        m = m + 1
    else
        Recovery1: Caused by MTU
        if P_ssh[i] < (Q between [s[i], s[i] + S_t]) then
            Calculate the difference J between P_ssh[i] and Q
            if 44x+28y == J then
                if x + y < o'[i] − o[i] then
                    Record o'[i]; s[i + 1] = o'[i]
                    m = m + 1
                    continue
                end if
            end if
        end if
        Recovery2: Caused by HTTP proxy filtering
        Remove 19 bytes from the raw HTTP symbol packet size of P_ssh[i] and recalculate P_ssh[i] by using the Algorithm 1
        if P_ssh[i] == (Q between [s[i], s[i] + S_t]) then
            Record o'[i]; s[i + 1] = o'[i]
            m = m + 1
        end if
    end if
end for
```

If the $i$th symbol packet cannot be found between $s_i$ and $s_i + S_t$, we use a recovery mechanism to identify symbols. Suppose that the symbol packet is combined with two types of packet fragments, 44 and 28 bytes. Given that the $i$th combined symbol packet size is $Q_j$ (smaller than 1,448 bytes), the location of $Q_j$ as $o'_i$, the original SSH packet size of the $i$th symbol is $P_{sshi}$, we first derive the combined packet fragment size $J = Q_j - P_{sshi}$. We then find integers $x$ and $y$ such that $J = 44x + 28y$. The sum of $x$ and $y$ is the total number of packets of 44 and 28 bytes generated during the packet interval $o'_i - o_i$. Therefore, we need to add the constraint $x + y < o'_i - o_i$ to find the $x$ and $y$ satisfying the equation $J = 44x + 28y$. If $x + y > o'_i - o_i$, we give up packet $Q_j$ and continue to find the next packet $Q_{j+1}$ in the threshold $S_t$. If the symbol packet is found, we set the real location as $o'_i$. To keep the false positive rate small, we need to carefully choose $S_t$. The smaller $S_t$ may fail to detect packets carrying symbols. The larger $S_t$ may erroneously pick packets that do not carry a symbol, but have the same size as the packet carrying symbols. As a result, the detection algorithm fails to recognize subsequent symbols.

We now address the issue of HTTP Proxy filtering. If a symbol SSH packet contains a HTTP header packet, the Anonymizer HTTP proxy will filter the HTTP field size of 19 bytes. To recover such a symbol packet, the size of the raw HTTP symbol packet $P_{Si}$ will subtract 19 bytes. Then, Algorithm 1 can be applied to recalculate the SSH packet size of $P_{Si}$ and search it in the threshold $S_t$ again.

### 4.7 Discussion

We now discuss the detectability of the attack and possible countermeasures.

#### 4.7.1 Attack Detectability

The enhanced covert channel attack is difficult to detect for the following reasons: 1) A successful attack needs to transmit only a short message known only to the attackers. As shown in Section 6.3, the attack only needs to manipulate 20 packets to achieve a desired high detection rate. 2) As shown in Fig. 2 in Section 2.2, the packet size observed at the client shows a large percentage of non-MTU packet size because of Anonymizer client software's buffering mechanism, manipulation, and Internet traffic dynamics. In a sequence of packets carrying a web object, there are a number of non-MTU packets randomly located in different places. Hence, it will be difficult for client to detect the attack by investigating whether non-MTU packets appear within the transmission of a web object. 3) To make the attack harder to be detected by anyone other than the attacker, we design various covert channel attacks to preserve the statistical properties of clean traffic (without message embedded) for modulated target traffic. First, we adopt the *Monte Carlo* sampling technique to ensure that the distribution of web packet size is preserved despite manipulation. Second, we develop techniques to pick up right packets of web objects to preserve the regularity of TCP packet size dynamics measured by *Hurst* parameter. Hence, client cannot detect the attack via measuring traffic dynamics.

#### 4.7.2 Countermeasures

In previous work [30], [31] the approach padding all packets to equal size can definitely defend against the attack proposed in this paper. Nevertheless, it is commonly believed that such all-out-effort padding will incur large overhead to both Anonymizer servers and the client, dramatically degrading the performance of anonymous communication [30]. To overcome the overhead, one possible alternative is selective padding similar to those in [2]. Nevertheless, it is an open question regarding the gains that Anonymizer can get and how much overhead it incurs to Anonymizer. We leave such investigation as our future work. Tor [32] may not resist our attacks either. Although Tor uses equal sized cells, it is only meaningful at the application player. An attacker may exploit the traffic dynamics and generate IP-layer packets of different sizes across the Tor network.

Besides padding all packets to equal size, we may consider other countermeasures such as creating packets of

different sizes randomly or much creating smaller packets. Nevertheless, the countermeasures may still leak information. In particular, the attacker may adapt the attack strategy by introducing extra delay interval between virtual web objects. Then, the attacker can distinguish the virtual web objects and accumulate the packet sizes of each object to derive the size of the virtual objects. Based on Shannon's perfect secrecy theory, if one can map any payload traffic to a predefined pattern, the attacker cannot obtain any information by analyzing the padded traffic. The padding strategy utilizes this principle and pads packets to equal size, which can be very expensive as we stated early.

A user may also use various cover traffic between users and Anonymizer [2], [33], [34] to counter the attack. For example, a user may simultaneously access a website while watching streaming videos from youtube. It will affect the accuracy of the attack to some extent. We will explore the effect of cover traffic on the accuracy of our attack in our future work.

#### 4.7.3 Comparison with Previous Work

We now illustrate the major difference between our work and representative related work [6]. Wang et al. [6] proposed a timing-based watermarking scheme by exploiting the inter-packet timing of a traffic flow. To make the watermarks reliable, their approach requires active web traffic lasting for 20 seconds to encode one signal bit. According to their real-world experiments over Anonymizer [6], to embed 32-bit watermarks into the traffic, their watermarking attack requires around 10 minutes of active web traffic. In addition, each packets of the web traffic will be delayed, which results in significant communication performance degradation. In comparison, our approach requires only tens of packets to achieve a high detection rate and low false positive rate. We would like to note that we only modulate 20 packets, which correspond to 7 seconds of web traffic, to achieve a detection rate around 100 percent. To summarize, our approach is more efficient and effective in comparison with the existing representative approach [6].

## 5 Analysis

In this section, we analyze the performance of the covert channel attack based on least significant packets. We derive the detection rate and false positive rate formulas and investigate what factors impact the attack effectiveness.

### 5.1 Detection Rate

According to our empirical observations and analysis, one way delay between the user and malicious website can be approximately modeled to have a *Patero* distribution, which is a heavy-tailed distribution, as illustrated in Fig. 12. Although the approximation is coarse, the analytical results reflect the essence of the attack.

The probability density function (PDF) of a Pareto distribution is given below:

$$f_X(x) = \begin{cases} \alpha \dfrac{x_m^\alpha}{x^{\alpha+1}}, & x > x_m, \\ 0, & x < x_m, \end{cases} \tag{24}$$

and the CDF of the Pareto distribution is

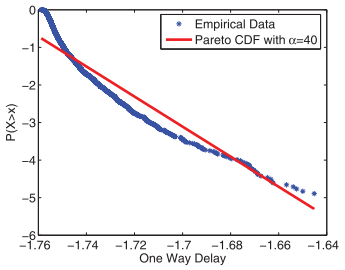


Fig. 12. Log-log plot of cumulative distribution $P(X > x)$ of one way delay.

$$F_X(x) = \begin{cases} 1 - \left(\dfrac{x_m}{x}\right)^\alpha, & x \geq x_m, \\ 0, & x < x_m. \end{cases} \tag{25}$$

Therefore,

$$P(X > x) = 1 - F_X(x) = \begin{cases} \left(\dfrac{x_m}{x}\right)^\alpha, & x \geq x_m, \\ 0, & x < x_m. \end{cases} \tag{26}$$

Fig. 12 plots one way delay distribution $P(X > x)$ in a log-log scale. We can derive shape parameter $\alpha$ from this plot.

From Fig. 12, we can derive the following Theorem 1, which is also validated by real experiments in Section 6.

**Theorem 1.** *Detection rate $P_D$ of a symbol from a SSH packet sequence is monotonously increasing with the delay interval $\Delta t$ added by tracebacker after a symbol packet.*

**Proof.** Assume that the arrival time of the $i$th packet carrying a symbol at Anonymizer is $T_i$ and the arrival time of the successive normal packet is $T_{i+1}$. Assume that $T_i$ and $T_{i+1}$ are independent and identically distributed (i.i.d) (rough appropriation while matching real work observations from Fig. 12 well). If these packets are combined at Anonymizer, the tracebacker cannot detect the packet carrying the symbol. The probability of detection error can be roughly calculated as follows in cases of without intelligent detection algorithm, $P_E = P(T_i > T_{i+1} + \Delta t) = P(T_i - T_{i+1} > \Delta t)$, where $\Delta t$ is delay interval added by the attacker between $T_i$ and $T_{i+1}$. Let $A = T_i - T_{i+1}$, we have $P_E = P(A > \Delta t) = 1 - P(A \leq \Delta t)$.

The detection rate $P_D$ is defined as the probability that one symbol is correctly recognized. Then, we have $P_D = 1 - P_E = P(A \leq \Delta t)$.

Let $T_i \doteq X$ and $T_{i+1} \doteq Y$. We have $A = X - Y$. Because $T_i$ and $T_{i+1}$ are independently and identically distributed, $X$ and $Y$ are i.i.d as well. Since $F_A(a) = P(A \leq a) = P(X - Y \leq a) = \int_{x_m}^{+\infty}(\int_{x_m}^{a+y} f(x,y)dx)dy$, then

$$\begin{aligned} P(A \leq \Delta t) = F_A(\Delta t) &= \int_{x_m}^{+\infty}\left(\int_{x_m}^{\Delta t+y} f(x,y)dx\right)dy \\ &= \int_{x_m}^{+\infty} f(y)\left(\int_{x_m}^{\Delta t+y} f(x)dx\right)dy \\ &= \int_{x_m}^{+\infty} f(y)\left(1 - \left(\frac{x_m}{\Delta t + y}\right)^\alpha\right)dy \\ &= \int_{x_m}^{+\infty} f(y)dy - \int_{x_m}^{+\infty} f(y)\left(\frac{x_m}{\Delta t + y}\right)^\alpha dy \\ &= 1 - \int_{x_m}^{+\infty}\left(\alpha\frac{x_m^\alpha}{y^{\alpha+1}}\right)\left(\frac{x_m}{\Delta t + y}\right)^\alpha dy. \end{aligned}$$

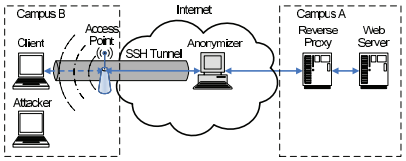


Fig. 13. Experiment setup.

We derive the first derivative of function $F_A(\Delta t)$ as

$$F'_A(\Delta t) = \int_{x_m}^{+\infty} \left(\alpha^2 \frac{x_m^{2\alpha}}{y^{\alpha+1}}\right)\left(\frac{1}{\Delta t + y}\right)^{\alpha+1} dy.$$

Since $x_m > 0$, $\Delta t > 0$, and $y > 0$, we have $F'_A > 0$. Therefore, $P_D > 0$ and $P_D$ is a monotonously increasing function in terms of $\Delta t$. The larger the delay interval an attacker chooses, the higher the detection rate. This analytical result is also validated by our real-world experimental data in Section 6. □

Detection rate $P_{D,n}$ is defined as the probability that the sequence of $n$-symbols embedded into target traffic at the malicious website is correctly recognized at the client. Given $P_D$, detection rate for 1-symbol message, $P_{D,n} = (P_D)^n$, which is also a monotonously increasing function with the delay interval.

### 5.2 False Positive Rate

When no symbols are embedded into target traffic, the detection algorithm could make a wrong decision. Let the probability of the least significant packets carrying the symbols be $q_1, q_2, \ldots, q_n$, where $n$ is the number of symbols. Denote the probability of the packet size MTU as $q_{mtu}$.

The false positive rate $P_{F,n}$ for detecting the sequence of $n$ symbols in normal traffic can be derived by

$$P_{F,n} = \prod_{i=1}^{n} q_i \leq (1 - q_{mtu})^n. \quad (27)$$

The inequality holds because $\sum q_i = 1 - q_{mtu}$.

Since the probability that a packet has a size of MTU is around 90 percent from our empirical data in Fig. 4, we have

$$P_{F,n} \leq (1 - q_{mtu})^n \approx \frac{1}{10^n}. \quad (28)$$

In fact, from Fig. 4, we know that the largest probability of a least significant packet is only 0.27 percent. Hence, the false positive is very low in practice. We will have a lower false positive rate when the original symbol length $n$ is large enough. Our extensive experimental results in Section 6 match this observation very well.

## 6 Evaluation over Anonymizer

In this section, we use real-world experiments to demonstrate the feasibility and effectiveness of the attacks. All the experiments were conducted in a controlled manner over the commercial Anonymizer and we experimented on TCP flows generated by ourselves to avoid legal issues.

### 6.1 Experiment Setup

Fig. 13 illustrates the experiment setup. We deploy a malicious website and a reverse proxy at City University of Hong Kong, denoted as Compus B. Campus A is the University of Massachusetts Lowell. The web server and reverse proxy are installed on a single computer. The web server is Apache/2.2.11 and reverse proxy is Pen [35]. Two other computers are deployed on Campus B. All computer are running Fedora Core 11 operating system. One computer acting as a client is connected to a wireless access point and its traffic is not encrypted over the network. The web browser is Firefox 3.5. We configure Firefox to not cache the data. The latest Adobe Flash plugin [36] is installed with the browser. The other computer is used as a sniffer to record the size of packets destined to client computer with source TCP port 22.

We modified the code of the reverse proxy *Pen* to control web packet size and implement the encoding algorithm. For the verification purpose, we downloaded the real-world webpages from CNN.com and deployed our own "CNN" web server to simulate a malicious website. To prevent other traffic from accessing our replicated website, we use the Linux firewall *iptables* and shutdown the website after our experiments were completed. By configuring the reverse proxy *Pen*, we map the reverse proxy port 8080 to the HTTP server port 80. Therefore, the reverse proxy can forward the packets for the web server. At the client side, the SSH client connects to the commercial Anonymizer server by the command *"ssh -L 80:cyberpass.net:80 username@cyberpass.net -N"* in the console with appropriate password. We configure the browser HTTP proxy as "127.0.0.1:80." Therefore, we use Firefox to browse the web server via the remote reverse proxy port "8080" and fetch the web objects via Anonymizer.

TABLE 1
*Hurst* Estimation Using R/S Method and Variance Method

| | R/S | | | Variance | | |
|---|---|---|---|---|---|---|
| Web Sites | O P S | L S P | M C S | O P S | L S P | M C S |
| cnn.com | 0.7949 | 0.8241 | 0.4990 | 0.7042 | 0.7594 | 0.4758 |
| nytimes.com | 0.8112 | 0.8874 | 0.5218 | 0.7473 | 0.7091 | 0.4804 |
| time.com | 0.8328 | 0.7595 | 0.5367 | 0.6964 | 0.7435 | 0.4679 |
| yahoo.com | 0.7743 | 0.8093 | 0.5280 | 0.7669 | 0.7756 | 0.5210 |
| nba.com | 0.7716 | 0.8699 | 0.5332 | 0.7154 | 0.7747 | 0.5231 |
| youtube.com | 0.9489 | 0.9791 | 0.5076 | 0.8099 | 0.7136 | 0.5133 |

### 6.2 TCP Packet Size Dynamics in Attacks

To illustrate the limitation of the simple attack in Section 3, we experiment both the R/S method and the variance method to estimate the *Hurst* value of the SSH packet size sequence. For demo purpose, we choose six popular websites and estimate their *Hurst* values. We consider three cases: 1) original packet size sequence (O P S), 2) the size sequence of packets carrying the covert channel attack based on least significant packets (L S P), and 3) the size sequence of packets carrying message based on Monte Carlo sampling (M C S) in the simple attack. Table 1 gives the *Hurst* values for above cases for each website. In addition, we select *cnn.com* and utilize a Matlab tool box (http://www.mathworks.com/matlabcentral/fileexchange/19148-hurst-parameter-estimate) to estimate the *Hurst* parameter and plot results from both R/S method and variance method. Figs. 14 and 17 illustrate the R/S plot and variance plot of the original packet size sequence. The Hurst parameter is much greater than 0.5. Therefore, the web packet size is self-similar. Figs. 15

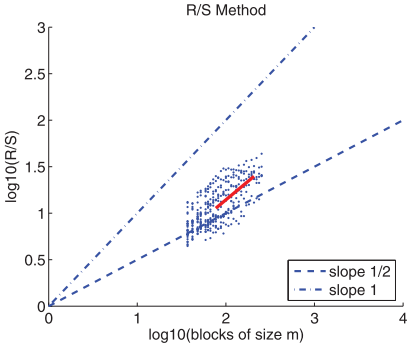


Fig. 14. R/S plot of the original packet sizes.

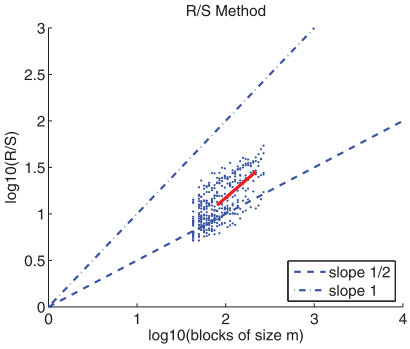


Fig. 15. R/S plot of the packet sizes with the covert channel attack based on least significant packets.

and 18 illustrate the R/S plot and variance plot of the packet size sequence with the covert channel attack based on the least significant packets. The Hurst parameter is also much greater than 0.5. The least significant packet based covert channel preserves web packet size self-similarity and is hard to detect. Figs. 16 and 19 illustrate the R/S plot and variance plot of the packet size sequence for the simple attack embedding the message based on Monte Carlo technique. It can be observed that Hurst parameter is around 0.5 and implies pure randomness. In summary, these figures and table show that the traffic embedded with a secret message using least significant packets is able to preserve the web TCP traffic self-similarity. Nevertheless, the *Hurst* parameter of around 0.5 for the traffic modulated in the simple attack indicates that the sequence of packet sizes is random. The simple attack is easier to detect because it cannot preserve the TCP dynamics.

### 6.3 Detection Rate

To validate the accuracy of the attack using least significant packets, we let the client browse our replicated webpages 30 times. At the reverse proxy, we generate a message and encrypt it with RC4 in counter mode. We then derive a sequence of symbols of length 20. Note that we generate HTTP objects of different size and calculate the location of the least significant packets, i.e. the location of our symbols. When the target web traffic arrives at the reverse proxy, we choose the symbol location, vary the read buffer, and embed the symbols into the target traffic. At the client side, Sniffer records the SSH packet size by removing the MAC (IEEE 802.11) header, IP header, and TCP header. Then, the detection algorithm proposed in Section 4.6 is used to recognize the symbols from the sequence of SSH packets.

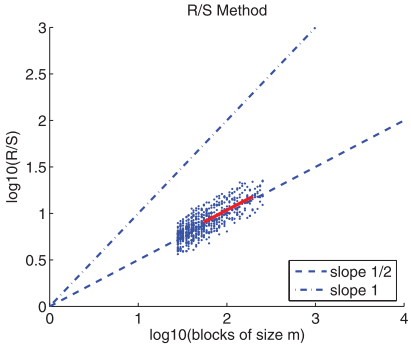


Fig. 16. R/S Plot of the packet sizes with the covert channel attack based on Monte Carlo sampling

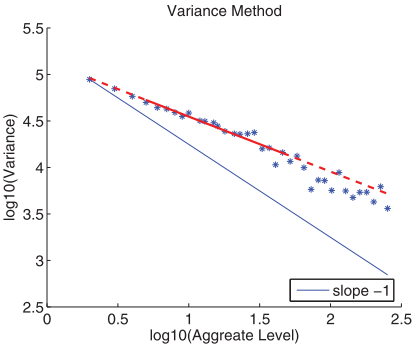


Fig. 17. Variance plot of the original packet sizes.

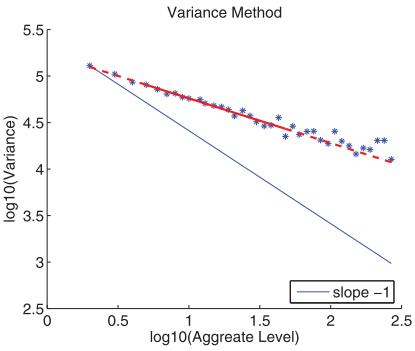


Fig. 18. Variance plot of the packet sizes with the covert channel attack based on least significant packets.

To evaluate the false positive rate of the attack, we let the client browse our replicated webpage 30 times via Anonymizer. No symbol is embedded into the traffic at the reverse proxy in these cases. We refer to the traffic without symbols as clean traffic. We then use the same detection algorithm proposed in Section 4.6 to detect 20 random symbols from clean traffic and calculate the false positive rate.

Fig. 20 illustrates the relationship between the detection rate and the delay interval, as well as the threshold $S_t$ in Algorithm 2. Recall $S_t$ is the threshold to restrain the difference between the real symbol packet position and its predefined position. From Fig. 20, we have a few observations. First, the best value of $S_t$ is around 8. This can be reasoned as follows: The smaller $S_t$ may not detect packets carrying symbols. The larger $S_t$ may erroneously pick packets that do not carry a symbol, but has the same size as the packet carrying a symbol. As a result, the detection algorithm cannot correctly recognize the later symbols. Second, the detection rate increases dramatically when the delay interval increases. This matches our analysis in

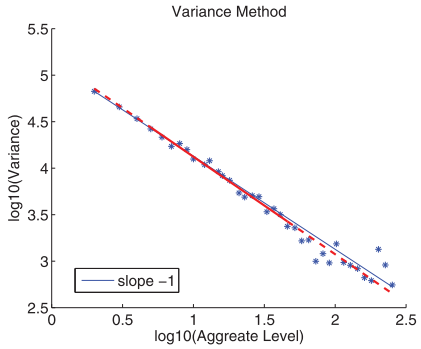


Fig. 19. Variance plot of the packet sizes with the covert channel attack based on Monte Carlo sampling.

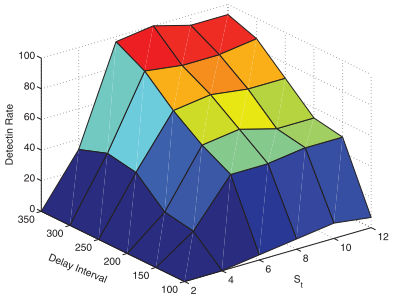


Fig. 20. Detection rate versus delay interval and $S_t$.

Section 5 very well. The detection rate approaches 100 percent when the delay interval is 350 ms and the threshold $S_t$ is 8. These results validate that the attack using least significant packets can significantly degrade the degree of anonymity service that Anonymizer promises.

Fig. 21 illustrates the relationship between the detection rate and the delay interval, as well as the number of symbols. Fig. 21 shows that the detection rate will decrease while the number of symbols increases, which is matched with our analysis in Section 5.1. From this figure, we know that only tens of packets is needed for our covert channel attack. This observation confirms that the attack is highly efficient and can compromise the anonymous web surfing very fast.

We did not plot the false positive rate since in all the cases, the false positive rate approaches 0. This matches with our analytical results in Section 5.2 very well.

## 7 Related Work

Traffic analysis and covert channel is a common means to degrade communication privacy. Existing traffic analysis can largely be categorized into two groups: passive traffic analysis and active watermarking techniques. Passive traffic analysis techniques have shown that the attacks record the traffic passively and identify the similarity between server's outbound traffic and client's inbound traffic [2], [3]. For example, Levineet al. [2] investigated a cross correlation technique for the similarity measurement. Other recent research have shown thatthe attackers can infer sensitive information from the encrypted network traffic by examining patterns in terms of the sizes of packet and its timing [11], [12], [14], [37]. For example, Sun et al. [11] investigated the sizes of the HTML objects transmitted over SSL connections and were able to identify the webpages based on the number and size of objects in each encrypted HTTP response.

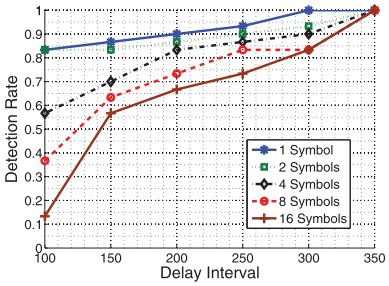


Fig. 21. Detection rate versus delay interval and number of symbols.

Liberatore and Levine [12] examined the packet sizes of HTTP traffic transmitted over persistent connection or tunneled via SSH port forwarding can statistically identify the webpages.

The active watermarking techniques intend to embed specific secret signal into the target traffic. Such techniques can reduce the false positive rate significantly if the signal is long enough and does not require massive training study of traffic cross correlation as required in passive traffic analysis [4], [6], [7], [10], [12], [13], [38], [39], [40]. For example, Peng et al. [41] analyzed the secrecy of timing-based watermarking technique proposed in [42], based on the distribution of traffic timing. Yu et al. [7] proposed a flow marking scheme based on the direct sequence spread spectrum (DSSS) technique. This technique could be used by attackers to secretly confirm the communication relationship via mix networks. Kiyavash et al. [38] proposed a multi-flow approach detecting the interval-based watermarks [6], [39] and DSSS-based watermarks [7]. Ling et al. [10] proposed the cell counter-based attack against Tor that the attackers embed a signal into the variation of cell counter of the target traffic. Wright et al. [40] proposed a traffic morphing approach that modulates the source packet sizes to resembles another one against the traffic classifiersfor VoIP [13] and web traffic [12]. Our covert attack is more efficient and effective than previous ones because it only needs to modulate 20 packets (7 seconds of web traffic) to achieve a detection rate around 100 percent.

Most existing work on covert channel focuses on timing-based techniques [21], [22], [43], [44]. For example, [22] exploited the statistical interpacket delays of legitimate network traffic and proposed a model-based covert timing channel to fit the modulated traffic behavior to legitimate traffic. Ramsbrock et al. [21] applied packet size based covert channel to Botnet and general network traffic.TCP packet size dynamics was not considered in their work. We are the first to apply packet size-based covert channel against Anonymizer service and explored various packet size distortion by Anonymizer proxies.

## 8 Conclusion

In this paper, we conducted a holistic investigation of *Anonymizer*, a known commercial anonymous communication system. We discovered the architecture of Anonymizer and found that the size of HTTP packets in the Anonymizer network is very dynamic. We investigated a class of novel covert channel attacks, which can drastically degrade the anonymity service provided by Anonymizer. We developed several salient techniques that make the attack efficient, accurate, and hard to detect. In particular, We applied

*Monte Carlo* sampling technique to carefully sample the target packet size ECDF to preserve packet size distribution. We designed techniques to choose right packets of web objects to preserve the regularity of TCP packet size dynamics measured by *Hurst* parameter. All these efforts make the attack practical and more undetectable. We also designed intelligent and robust detection algorithms to recover the distorted symbols caused by Anonymizer and Internet traffic dynamics. Extensive analysis and experiments were conducted to validate the effectiveness and feasibility of the proposed attacks. Our data show that the enhanced covert channel attack could dramatically and quickly degrade the anonymity service by Anonymizer. Defending against the proposed attacks remains a challenging task. We plan to work with Anonymizer developers and investigate the solution in our future work.

## ACKNOWLEDGMENTS

This work was supported in part by National Key Basic Research Program of China (973 Program) under Grants 2010CB328104 and 2011CB302800, the China National High Technology Research and Development Program under Grants No. 2013AA013503, National Science Foundation of China (NSFC) under Grants 61272054, 61202449, 61003257, 61320106007, 61070221, 61070222, the US National Science Foundation (NSF) under Grants CNS-1116644, DUE-0942113, CNS-0958477, CNS-1117175, CNS-0916584, CNS-1065136, CNS-1218876, and by General Research Fund of the Hong Kong SAR, China No. (CityU 114012, CityU 114513), ShenZhen (China) Basic Research Project No. JCYJ20120618115257259, by China Specialized Research Fund for the Doctoral Program of Higher Education under Grant 20110092130002, JScience Research Foundation of Graduate School of Southeast University, Jiangsu Provincial Key Laboratory of Network and Information Security under Grant BM2003201, and Key Laboratory of Computer Network and Information Integration of Ministry of Education of China under Grant 93K-9. Any opinions, findings, conclusions, and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

## REFERENCES

[1] L. Overlier and P. Syverson, "Locating Hidden Servers," *Proc. IEEE Security and Privacy Symp. (S&P),* May 2006.

[2] B.N. Levine, M.K. Reiter, C. Wang, and M. Wright, "Timing Attacks in Low-Latency Mix-Based Systems," *Proc. Eighth Int'l Financial Cryptography (FC) Conf.,* Feb. 2004.

[3] Y. Zhu, X. Fu, B. Graham, R. Bettati, and W. Zhao, "On Flow Correlation Attacks and Countermeasures in Mix Networks," *Proc. Workshop Privacy Enhancing Technologies (PET),* May 2004.

[4] S.J. Murdoch and G. Danezis, "Low-Cost Traffic Analysis of Tor," *Proc. IEEE Security and Privacy Symp. (S&P),* May 2006.

[5] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker, "Low-Resource Routing Attacks against Anonymous Systems," *Proc. ACM Workshop Privacy Electronic Soc. (WPES),* Oct. 2007.

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

[7] 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 (S&P),* May 2007.

[8] A. Houmansadr, N. Kiyavash, and N. Borisov, "Rainbow: A Robust and Invisible Non-Blind Watermark for Network Flows," *Proc. 16th Network and Distributed System Security Symp. (NDSS),* Feb. 2009.

[9] N. Evans, R. Dingledine, and C. Grothoff, "A Practical Congestion Attack on Tor Using Long Paths," *Proc. 18th USENIX Security Symp.(Security),* Aug. 2009.

[10] 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),* Nov. 2009.

[11] Q.X. Sun, D.R. Simon, Y. Wang, W. Russell, V.N. Padmanabhan, and L.L. Qiu, "Statistical Identification of Encrypted Web Browsing Traffic," *Proc. IEEE Symp. Security and Privacy (S&P),* May 2002.

[12] M. Liberatore and B.N. Levine, "Inferring the Source of Encrypted HTTP Connections," *Proc. ACM Conf. Computer and Comm. Security (CCS),* Oct. 2006.

[13] C.V. Wright, L. Ballard, S.E. Coull, F. Monrose, and G.M. Masson, "Language Identification of Encrypted VOIP Traffic: Alejandra y Roberto or Alice and Bob," *Proc. 16th Ann. USENIX Security Symp. (Security),* Aug. 2007.

[14] C.V. Wright, L. Ballard, S.E. Coull, F. Monrose, and G.M. Masson, "Spot Me If You Can: Uncovering Spoken Phrases in Encrypted VOIPConversation," *Proc. IEEE Symp. Security and Privacy (S&P),* May 2008.

[15] W.E. Leland, M.S. Taqqu, W. Willinger, and D.V. Wilson, "On the Self-Similar Nature of Ethernet Traffic (Extended)," *IEEE/ACM Trans. Networking,* vol. 2, no. 1, pp. 1-15, Feb. 1994.

[16] J. Beran, *Statistics for Long-Memory Processes.* Chapman & Hall, Oct. 1994.

[17] Anonymizer, Inc., http://www.anonymizer.com/, 2011.

[18] T. Ylonen and C. Lonvick, "The Secure Shell (Ssh) Transport Layer Protocol, RFC 4253," http://www.ietf.org/rfc/rfc4253.txt, Jan. 2006.

[19] T. Ylonen and C. Lonvick, "The Secure Shell (SSH) Authentication Protocol, RFC 4252," http://www.ietf.org/rfc/rfc4252.txt, Jan. 2006.

[20] T. Ylonen and C. Lonvick, "The Secure Shell (SSH) Connection Protocol, RFC 4254," http://www.ietf.org/rfc/rfc4254.txt, Jan. 2006.

[21] D. Ramsbrock, X. Wang, and X. Jiang, "A First Step Towards Live Botmaster Traceback," *Proc. 11th Int'l Symp. Recent Advances in Intrusion Detection (RAID),* Sept. 2008.

[22] 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),* Sept. 2008.

[23] H.E. Hurst, "Long Term Storage Capacity of Reservoirs," *Trans. Am. Soc. Civil Engineers,* vol. 116, pp. 770-799, 1951.

[24] R.G. Clegg, "A Practical Guide to Measuring the Hurst Parameter," *Proc. 21st UK Performance Eng. Workshop,* 2005.

[25] B.B. Mandelbrot and J.W.V. Ness, "Fractional Brownian Motions, Fractional Noises and Applications," *Soc. Industrial and Applied Math.,* vol. 10, no. 4, pp. 422-437, Oct. 1968.

[26] I. Cox, M. Miller, J. Bloom, J. Fridrich, and T. Kalker, *Digital Watermarking and Steganography,* second ed. Morgan Kaufmann, 2007.

[27] J. Soto and L. Bassham, "Randomness Testing of the Advanced Encryption Standard Finalist Candidates," *NIST IR 6483, Nat'l Inst. of Standards and Technology,* 1999.

[28] H.-K. Choi and J.O. Limb, A Behavioral Model of Web Traffic," *Proc. IEEE Int'l Conf. Network Protocols (ICNP),* Sept. 1999.

[29] J.J. Lee and M. Gupta, "A New Traffic Model for Current User Web Browsing Behavior," technical report, Intel Corp., Santa Clara, Calif, 2007.

[30] X. Fu, B. Graham, Y. Guan, R. Bettati, and W. Zhao, "NetCamo: Camouflaging Network Traffic for Real-Time Applications," *Proc. Texas Workshop Security of Information Systems,* Apr. 2003.

[31] W. Dai, "Pipenet 1.1," http://weidai.com/pipenet.txt, 2011.

[32] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The Secondgeneration Onion Router," *Proc. 13th USENIX Security Symp.,* Aug. 2004.

[33] X. Fu, B. Graham, R. Bettati, and W. Zhao, "Analytical and Empirical Analysis of Countermeasures to Traffic Analysis Attacks," *Proc. Int'l Conf. Parallel Processing (ICPP),* 2003.

[34] V. Shmatikov and M. hsiu Wang, "Timing Analysis in Low-Latency Mix Networks: Attacks and Defenses," *Proc. European Symp. Research in Computer Security (ESORICS),* 2006.

[35] "Pen," http://siag.nu/pen, 2011.

[36] "Adobe Flash Player," http://www.adobe.com/products/flashplayer/, 2011.

[37] D.X. Song, D. Wagner, and X. Tian, "Timing Analysis of Keystrokes and Timing Attacks on SSH," *Proc. 10th USENIX Security Symp.,* Aug. 2001.
[38] N. Kiyavash, A. Houmansadr, and N. Borisov, "Multi-Flow Attacks against Network Flow Watermarking Schemes," *Proc. 17th USENIX Security Symp.,* July/Aug. 2008.
[39] Y.J. Pyun, Y.H. Park, X. Wang, D.S. Reeves, and P. Ning, "Tracing Traffic through Intermediate Hosts that Repacketize Flows," *Proc. IEEE INFOCOM,* May 2007.
[40] C.V. Wright, S.E. Coull, and F. Monrose, "Traffic Morphing: An Efficient Defense against Statistical Traffic Analysis," *Proc. Network and Distributed Security Symp. (NDSS),* Feb. 2009.
[41] P. Peng, P. Ning, and D.S. Reeves, "On the Secrecy of Timing-Based Active Watermarking Trace-Back Techniques," *Proc. IEEE Security and Privacy Symp. (S&P),* May 2006.
[42] X. Wang and D.S. Reeves, "Robust Correlation of Encrypted Attack Traffic through Stepping Stones by Manipulation of Inter-Packet Delays," *Proc. ACM Conf. Computer and Comm. Security (CCS),* Nov. 2003.
[43] G. Shah, A. Molina, and M. Blaze, "Keyboards and Covert Channels," *Proc. 15th USENIX Security Symp.,* July/Aug. 2006.
[44] S. Cabuk, C.E. Brodley, and C. Shields, "IP Covert Timing Channels: Design and Detection," *Proc. ACM Conf. Computer and Comm. Security (CCS),* Oct. 2004.

**Zhen Ling** received the BS degree in computer science from Nanjing Institute of Technology, China, in 2005, and is working toward the PhD degree in the School of Computer Science and Engineering, Southeast University, Nanjing, China. He joined Department of Computer Science at the City University of Hong Kong from 2008 to 2009 as a research associate, and then joined Department of Computer Science at the University of Victoria from 2011-2013 as a visiting scholar. His research interests include network security, privacy, and forensics. He is a member of the IEEE.

**Xinwen Fu** received the BS and MS degrees in electrical engineering in 1995 and 1998, respectively, from Xi'an Jiaotong University, China, and the University of Science and Technology of China, respectively. He received the PhD degree in 2005 in computer engineering from Texas A&M University. He is an associate professor in the Department of Computer Science, University of Massachusetts Lowell. From 2005 to 2008, he was an assistant professor with the College of Business and Information Systems at Dakota State University. In the summer of 2008, he joined University of Massachusetts Lowell as a faculty member. His current research interests include network security and privacy. He is a member of the IEEE.

**Weijia Jia** received the BSc and MSc degrees from Center South University, China, in 1982 and 1984, respectively, and the master's of applied sciences and the PhD degree from Polytechnic Faculty of Mons, Belgium in 1992 and 1993, respectively, all in computer science. He is currently a full professor in the Department of Computer Science and the director of Future Networking Center, ShenZhen Research Institute of City University of Hong Kong (CityU). He joined German National Research Center for Information Science (GMD) in Bonn (St. Augustine) from 1993 to 1995 as a research fellow. In 1995, he joined Department of Computer Science, CityU as an assistant professor. His research interests include next generation wireless communication, protocols and heterogeneous networks; distributed systems, multicast and anycast QoS routing protocols.

**Wei Yu** received the BS degree in electrical engineering from Nanjing University of Technology in 1992, the MS degree in electrical engineering from Tongji University in 1995, and the PhD degree in computer engineering from Texas A&M University in 2008. He is an assistant professor in the Department of Computer and Information Sciences, Towson University, Maryland. Before that, he was with Cisco Systems Inc. for almost nine years. His research interests include cyber space security, computer network, and distributed systems. He is a member of the IEEE and the IEEE Computer Society.

**Dong Xuan** received the BS and MS degrees in electronic engineering from Shanghai Jiao Tong University (SJTU), China, in 1990 and 1993, and the PhD degree in computer engineering from Texas A&M University in 2001. Currently, he is a full professor in the Department of Computer Science and Engineering, The Ohio State University (OSU). He was with the faculty of Electronic Engineering at SJTU from 1993 to 1998. His research interests include distributed computing, computer networks, and cyberspace security. He received the US National Science Foundation (NSF) CAREER Award in 2005 and the Lumley Research Award from the College of Engineering, OSU in 2009. He is a member of the IEEE.

**Junzhou Luo** received the BS degree in applied mathematics from Southeast University in 1982, and the MS and PhD degrees in computer network both from Southeast University in 1992 and in 2000, respectively. He is a full professor in the School of Computer Science and Engineering, Southeast University, Nanjing, China. His research interests include next generation network, protocol engineering, network security and management, grid and cloud computing, and wireless LAN. He is a member of the IEEE and the IEEE Computer Society.