SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks 2013; **6**:257–274
Published online 17 May 2012 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/sec.540

RESEARCH ARTICLE

Blind detection of spread spectrum flow watermarks†

Weijia Jia[1], Fung Po Tso[1], Zhen Ling[2], Xinwen Fu[3], Dong Xuan[4] and Wei Yu[5]*

[1] City University of Hong Kong, Tat Chee Avenue, Kowloon, Hong Kong
[2] School of Computer Science and Engineering, Southeast University, Liwenzheng Building (North) #241, Nanjing 210096, China
[3] Department of Computer Science, University of Massachusetts Lowell, Lowell, MA 01854, U.S.A.
[4] Department of Computer Science and Engineering, The Ohio State University, Columbus, OH 43210, U.S.A.
[5] Department of Computer and Information Sciences, Towson University, Towson, MD, U.S.A

ABSTRACT

Recently, the direct sequence spread spectrum (DSSS)-based technique has been proposed to trace anonymous network flows. In this technique, homogeneous pseudo-noise (PN) codes are used to modulate multiple bit signals that are embedded into the target flow as watermarks. This technique could be maliciously used to degrade an anonymous communication network. In this paper, we propose an effective single flow-based scheme to detect the existence of these watermarks. Our investigation shows that, even if we have no knowledge of the applied PN code, we are still able to detect malicious DSSS watermarks via mean-square autocorrelation (MSAC) of a single modulated flow's traffic rate time series. MSAC shows periodic peaks because of self-similarity in the modulated traffic caused by homogeneous PN codes that are used in modulating multiple bit signals. Our scheme has low complexity and does not require any PN code synchronization. We evaluate this detection scheme's effectiveness via simulations. Our results demonstrate a high detection rate with a low false positive rate. Real-world experiments on Tor also validate the feasibility of the detection scheme. Our scheme is more flexible and accurate than the existing multiflow-based approach in DSSS watermark detection. We also present a theory for reconstructing the DSSS code once the DSSS code length is known and simulations validate the feasibility.

KEYWORDS
anonymity; detection; DSSS; mean-square autocorrelation

***Correspondence**
Wei Yu, Department of Computer and Information Sciences, Towson University, Towson, MD 21252, U.S.A.
E-mail: wyu@towson.edu

†A preliminary version of this paper has appeared in the Proceedings of the IEEE Conference on Computer Communications (INFOCOM), April 2009.

1. INTRODUCTION

In recent years, significant progress has been made in the journey of fighting against attacks on anonymous communication networks. However, the journey is far from over [1–6]. Significant research efforts are still needed to discover and then defend against these attacks.

A new type of attack against anonymous communication networks has been discovered recently. The attack is based on the spread spectrum (SS) communication technique. In [7], Yu *et al*. proposed a direct sequence spread spectrum (DHSS)-based traceable technique. In this technique, at the sender side, homogeneous pseudo-noise (PN) codes are used to modulate multiple bit signals that are embedded into the target flow as watermarks. The applied PN codes, independent of the original data signal, 'spread' the signal during transmission. At the receiver side, the signal is recovered ('despread') on the basis of the same PN code. Spread spectrum techniques are resistant to interference and interception. This technique could be maliciously used by an attacker in order to trace users of an anonymous communication network. In such an attack, the attacker, called *interferer*, can modulate a victim sender's outbound traffic flow rate via a secret PN code, analogous to watermarking the target traffic. Another attacker, called *sniffer*, can recover the signal from a victim receiver's inbound traffic flow on the basis of the secret PN code. In this way, the attackers can confirm the communication relationship between the sender and the receiver.

The above attack is difficult to detect because DHSS-modulated traffic shows a white noise-like pattern both in the frequency domain and time domain. Some efforts have been made to detect such attacks. In [8], Kiyavash *et al*. introduced a multiflow approach to detect DHSS watermarks. This approach requires a Markov-modulated Poisson process (MMPP) model.

In this paper, we introduce a statistical approach in detecting DHSS watermarks and defeating malicious traceable. In

the DHSS watermarking scheme, a single PN code is used to spread multiple bits of a signal. Traffic, with such a spread-signal embedded, demonstrates self-similarity. A sender or a receiver, suspicious about being traced, may use the *mean-square autocorrelation* (MSAC) of a traffic rate time series to detect such self-similarity: MSAC applied to traffic marked with such a signal demonstrates periodicity, showing peaks at regular intervals. This will expose the malicious traceable activity. We conduct a thorough theoretic analysis of this approach in detecting DHSS watermarks. We also study the approach to accurately recover the DHSS code embedded in marked traffic. We evaluate the effectiveness of the detection scheme via simulations and experiments on Tor, a real-world anonymous communication system. Our results demonstrate a high detection rate with a low false positive rate. Our scheme has low complexity and does not require any PN code synchronization. Our approach in detecting malicious DHSS watermarks is simple, efficient, and effective, compared with the approach in [8]. Our approach can deal with a single flow (or a few flows), whereas their approach does not. Our approach is based on a simple statistic, mean-square autocorrelation, whereas their approach requires a MMPP model.

Once the malicious traceable has been detected, the victim sender or receiver may stop the communication to thwart further detection. We also discuss the approach that enables the reconstruction of the DHSS code.[‡] If the DHSS code is reconstructed, the sender and receiver may introduce marks into other traffic, increase the detection positive rate, and mislead the malicious traceable. In addition, the sender or the receiver may also initiate the process of network forensics to locate the attacker. In this paper, we focus on detecting malicious traceable. The part of the intrusion reaction forensics is not within the scope of this paper.

The rest of the paper is organized as follows. In Section 3, we review the DHSS-based traceable technique introduced in [7]. In Section 4, we introduce the MSAC and our approach in detecting a malicious DHSS-based traceable by calculating MSAC of the DHSS-modulated traffic. We also discuss various parameters that affect the detection of DHSS-modulated traffic and the approach in reconstructing the DHSS code. In Section 5, we use ns-2 simulations and network experiments over *Tor* to validate our findings, respectively. The disadvantage of the multiflow detection approach in [8] is also presented in Section 5.5. We review related work in Section 2 and conclude this paper in Section 6.

2. RELATED WORK

Chaum pioneered the idea of anonymous communication systems in [9]. A good review of various mix systems can be found in [10,11]. There has been much research on degrading anonymous communication through mix networks. In order to determine whether Alice is communicating with Bob, through a mix network, the similarity between Alice's outbound traffic and Bob's inbound traffic may be measured. For example, Zhu *et al*. in [12] proposed the scheme of using mutual information for the similarity measurement. Levine *et al*. in [13] utilized a cross-correlation technique. Murdoch *et al*., in [4], also investigated the timing-based threats on Tor by using some compromised Tor nodes. Fu *et al*. [14] studied a flow-marking scheme. Overlier *et al*. [5] studied a scheme using one compromised mix node to identify the 'hidden server' anonymized by Tor. Yu *et al*. [7] proposed a direct sequence spread spectrum (DHSS)-based traceable technique, which could be maliciously used to trace users of an anonymous communication network. In this technique, attackers modulate a victim's traffic flow using a secret PN code.

Interval-based watermarks are proposed to trace attackers through the stepping stones. Wang *et al*. in [15] proposed a scheme that injected nondisplayable content into packets. Wang *et al*. in [16] proposed an active watermarking scheme that was robust to random timing perturbation. They analyzed the tradeoffs between the true positive rate, the maximum timing perturbation added by attackers, and the number of packets needed to successfully decode the watermark. Wang *et al*. in [17] also investigated the feasibility of a timing-based watermarking scheme in identifying the encrypted peer-to-peer VoIP calls. By slightly changing the timing of the packets, their approach can correlate encrypted network connections. Nevertheless, these timing-based schemes are not effective in tracing communication through a mix network with batching strategies that manipulate interpacket delivery timing, as indicated in [7]. Peng *et al*. in [18] analyzed the secrecy of timing-based watermarking traceable proposed in [16], on the basis of the distribution of traffic timing. Nevertheless, our focus in this paper is to detect the malicious DHSS-based flow-marking technique proposed in [7].

Kiyavash *et al*. [8] proposed a multiflow approach in detecting the interval-based watermarks (which modify packet timings by selectively delaying some packets [19,2]) and DHSS watermarks [7]. This approach requires multiple watermarked flows, which may show an unusual long silence period without packets or an unusual long period of low-rate traffic. They also proposed approaches in recovering the watermarking parameters and removing watermarks in the case of interval-based watermarks. They applied a probabilistic model and the MMPP to demonstrate the principle of their approaches. The authors briefly discussed countermeasures, which require more in-depth discussion. Note that, given so many flows over the Internet, it is not always easy to recognize and find a relatively large number of flows embedded with DHSS marks. Although the multiple flow attack is feasible in theory, its usage in practice needs further investigation.

3. BACKGROUND

In this section, we first review the basic framework of the DHSS watermarking technique and then discuss the secrecy

[‡]We use the DHSS code and PN code interchangeably in this paper.

DOI: 10.1002/sec

of this technique on how to escape detection. An introduction to the basic DHSS principle can be found in Appendix A.

3.1. Framework of DHSS watermarking

The framework for the DHSS watermarking technique in [7] is illustrated in Figure 1. The basic idea is illustrated here The malicious *interferer* first spreads each bit of a signal through a secret PN code, and the spread signal is used to modulate the target traffic rate so that the signal is embedded into the target traffic initiated by a *victim sender*. The *sniffer*, at a victim *receiver*'s side, extracts the spread signal from the target traffic via a digital filter, and the same PN code is used in despreading and recovering the original signal. If the original signal is recovered by the sniffer, the communication relationship between the sender and receiver is confirmed. There are two important modules within the framework: (i) mark generation at the interferer and (ii) mark recognition at the sniffer.

Mark generation module at the interferer:

(1) An original signal bit x of '+1' or '−1' is to be transmitted (to transmit a w-bit signal, just repeat the following steps). This original signal will normally consist of multiple bits because longer signals decrease the false positive rate for a traceable [7]. The transmitted baseband signal X can be written as

$$X = xC_t, \tag{1}$$

where C_t is a PN code with chip duration t_c.

(2) X is then used to modulate a victim traffic flow. When a chip of X is -1, *strong interference* is applied against the flow so that the flow has a lower rate for t_c seconds. When a chip is $+1$, *weak interference* (or no interference) is applied against the flow so that the flow has a higher rate for t_c seconds. If the flow has an average rate of D, then the high rate is $D+A$, and the low rate is $D-A$, where A is the *mark amplitude*. The rate of the target traffic flow must be large enough for the adversarial interferer to introduce marks. The transmitted signal T_x (also called *DHSS watermarks* or PN code-modulated traffic) can be represented by

$$T_x = AxC_t + D \tag{2}$$

(3) The modulated flow is transmitted via the Internet, where noise can be introduced by cross traffic. All noise is treated as an aggregated factor.

Mark recognition module at the sniffer:

(1) By denoting noise as a random variable ξ, we can formulate the received signal R_x as,

$$R_x = AxC_t + D + \xi \tag{3}$$

A *sniffer* derives R_x by capturing a traffic segment at the victim receiver then dividing it into chunks. Each chunk lasts for a chip duration of t_c seconds, and the average traffic rate of each chunk can then be calculated. The average rate for l continuous chunks constitutes R_x. All items in Equation (3) are $1 \times l$ vectors, where l is the PN code length, that is the number of chips in a PN code.

(2) A high-pass filter is applied against the received signal R_x in order to remove the direct current component D from the received signal. Then, the filtered received signal R_x can roughly be represented as follows

$$R_x \approx AxC_t + \xi \tag{4}$$

(3) A locally generated PN code C_r, the same as the code at the interferer, is used to despread the filtered received signal R_x to derive the received baseband signal R_b,

$$R_b = R_x \cdot C_r = AdtC_t \cdot C_r + \xi \cdot C_r \tag{5}$$

where $\cdot$ refers to the *dot product* operation. When $C_r = C_t$, the signal can be recovered.

(4) Then, a simple decision rule classifies the received signal (or bit) as $+1$ or -1.

In practice, in order to recognize the original signal x, DHSS watermarking requires that the locally generated PN code at the sniffer is synchronized with the one at the interferer. In order to address this problem, a matched filter-based approach is proposed in [7]. As indicated in [7], PN code length, original signal length, chip duration t_c, and mark amplitude A all affect traceable performance.

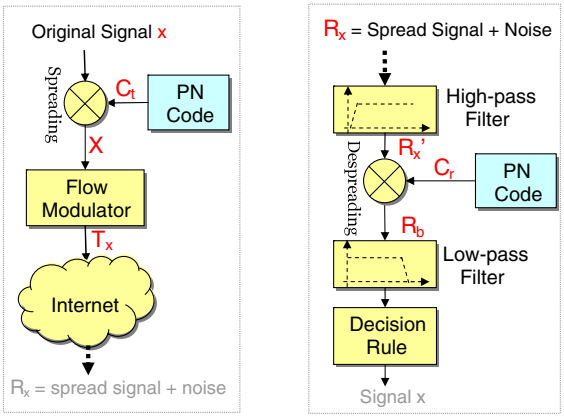

Figure 1. Framework of direct sequence spread spectrum-based traceable. (a) Interferer and (b) sniffer.

There are mature PN code generators such as *m-sequences code*, *Barker code*, *gold codes*, and *Hadamard–Walsh codes* [20,21] that may be used. The work in [7] used the m-sequence code, which has a sharp autocorrelation function [20]. This characteristic makes it easier for the *sniffer* to accurately synchronize and recognize watermarks in the target traffic.

3.2. Secrecy of DHSS watermarking

Secrecy of the DHSS watermarks refers to the difficulty of detecting the traceable. It is desired by attackers trying to escape detection. The DHSS-based traceable uses the following mechanisms for secrecy: (i) *secrecy of the code*. The DHSS watermarks provide secrecy on the basis of the secrecy of the code (including the chip duration). Because we do not know the code, it is difficult to recover the embedded signal. However, our primary goal in this paper is to detect the fact of the malicious traceable. (ii) *Low mark aptitude*. A carefully chosen mark amplitude A in Equation (2) can be very small in comparison with noise so that the DHSS mark X is covered by the noise ξ in the received signal R_x. The recognition process will effectively restore the spread signal to its narrow band and recover the original signal x from the noise. (iii) *DHSS watermarks show a white noise-like pattern in both time and frequency domains*. PN code-modulated traffic appears random for those who do not know the code. In general, the greater the code length, the harder the code is to detect. The signal x is also designed to appear randomly in order to maintain the secrecy. It is not feasible to recognize PN code-modulated traffic in both time and frequency domains.

4. BLIND DETECTION OF DHSS WATERMARKS

In this section, we introduce an approach to blindly detect DHSS watermarks. The detection process can be conducted by a sender or a receiver suspicious of being maliciously traced. After detection, the sender and receiver may stop communication to thwart such malicious traceable. Although it is difficult to recognize the presence of PN code-modulated traffic by searching for patterns in the time or frequency domains, we demonstrate other features that can reveal the existence of marked traffic blindly, without any knowledge of the applied PN code. We investigate and apply the MSAC, which measures the similarity of a PN code-modulated traffic segment and a time-shifted version of the same segment. The inspiration for this comes from the fact that the same PN code is used to spread each bit of a signal. If we can synchronize a time-shifted segment of the traffic with the original segment, the PN code reinforces rather than cancels, and an observable pattern emerges. In addition, we discuss an effective approach that enables the reconstruction of the DHSS code and can be used in misleading the malicious traceable.

To simplify our analysis, we assume that the chip duration is 1 unit (e.g., 1 s), unless explicitly stated. We will analyze the MSAC of DHSS watermarks in the synchronized case, where a traffic segment, a *window*, begins at a bit boundary and contains complete spread bits; in this case, the window is a multiple of l, where l is the PN code length. Then, we will analyze the MSAC in the nonsynchronized case, where a window doesn't necessarily begin (or end) on a bit boundary. Finally, we describe the workflow in detecting DHSS watermarks, introduce an automatic decision rule, and discuss related issues.

4.1. Mean-square autocorrelation in a synchronized window

Recall that our objective is to determine whether the traffic ismodulated by a PN code or not. Denote $\rightarrow x = \{x_0, \cdots, x_{w-1}\}$ as the signal, a series of bits, where the number of bits w is the *window* size. Therefore, a window contains w complete bits. Denote a PN code as $\rightarrow C = \{c_0, \cdots, c_{l-1}\}$, where l is the code length. We assume that bits x_i and x_j $(i \neq j)$ are independent because it is the worst case in detecting DHSS watermarks. Cases where bits are not independent actually facilitate PN code detection; for example, a modulated signal of all 1 s clearly has a period of length l and may demonstrate peaks in the frequency domain.

Therefore, the modulated signal $\rightarrow X$ can be written as follows:

$$\begin{aligned} \rightarrow \mathbf{X} &= (x_0 \rightarrow C, x_1 \rightarrow C, \cdots, x_{w-1} \rightarrow C), \quad (6)\\ &= (x_0 c_0, \cdots, x_0 c_{l-1}, \cdots, \\ &\quad x_{w-1} c_0, \cdots, x_{w-1} c_{l-1}) \quad (7) \end{aligned}$$

where $\rightarrow \mathbf{X}$ is a vector of length wl. In (7), c_j is one chip of a PN code, and $c_j = 1$ or -1. x_i is one bit of the signal and $x_i = A$ or $-A$, where A is the mark amplitude. We assume that x_i $(0 \leq i \leq w-1)$ is independent and identically distributed. Therefore, $Pr(x_i c_j = A) = 1/2$, $Pr(x_i c_j = -A) = 1/2$, so that $E(x_i c_j) = 0$ and the standard deviation $\sigma = A$. We can use Equation (8) to estimate the autocorrelation of a time series represented by $\rightarrow \mathbf{X}$

$$r(\tau) = \frac{1}{(wl - \tau)} \sum_{i=1}^{wl-\tau} y_i y_{i+\tau} \quad (8)$$

In Equation (8), τ is the lag, and $y_i = x_{\lfloor i/l \rfloor} c_{i\%l}$ is the ith item of $\rightarrow \mathbf{X}$, where $\lfloor i/l \rfloor$ is the quotient of i divided by l and $i\%l$ is the remainder. Here is a special case of $r(\tau)$. When $\tau = kl$, from Equations (7) and (8),

$$\begin{aligned} r(kl) &= \frac{1}{wl - kl} \sum_{i=0}^{wl-kl-1} y_i y_{i+kl} \quad (9)\\ &= \frac{1}{wl - kl} (x_0 x_{0+k} \rightarrow C \cdot \rightarrow C \\ &\quad + \cdots + x_{w-1-k} x_{w-1} \rightarrow C \cdot \rightarrow C) \quad (10) \end{aligned}$$

where $\cdot$ refers to *dot product* and $\rightarrow C \cdot \rightarrow C = l$. Therefore,

$$r(kl) = \frac{1}{w-k}(x_0 x_{0+k} + \cdots + x_{w-1-k} x_{w-1}) \tag{11}$$

$r^2(\tau)$ is the square autocorrelation of the spread signal $\rightarrow \mathbf{X}$ and a time-shifted $\rightarrow \mathbf{X}$ with lag τ. For $r^2(\tau)$, we have Theorem 1, suggested by the following reasoning. Recall that, to recognize the watermarks, the adversary initiating the malicious traceable needs to know the original code. Because we do not have access to the code, we cannot retrieve the watermarks. However, the PN code sequence is used repeatedly in a long signal, and we can compare one part of a signal with a later part of the same signal. Because the PN code is embedded repeatedly in the signal, we can attempt to align time-shifted portions of the signal with itself. Normally, this reveals nothing because of the autocorrelation property of PN codes. However, when the sequence is shifted by a precise multiple of the PN code length and correlated, the autocorrelation yields a nonzero result (which can be positive or negative). Such results can still be obscured by noise. Squaring such results can guarantee that the value is positive. We then repeat the calculation for multiple segments, sum the results, and calculate the average. The square autocorrelation of different segments will reinforce each other so that peaks emerge at multiples of l within the mean-square autocorrelation. This self-similarity reveals the presence of DHSS watermarks.

Theorem 1. $E(r^2(\tau))$ *demonstrates periodicity* with τ $(0 \le \tau < wl)$,

$$E\left(r^2(\tau)\right) \approx \begin{cases} A^4, & \tau = 0, (12) \\ \dfrac{A^4}{w-k}, & \tau = kl, 0 < k < w, (13) \\ 0, & \tau \neq kl, 0 < k < w \end{cases} \tag{12}$$

The proof of Theorem 1 can be found in Appendix B. The proof demonstrates the key reason why the PN code-modulated traffic can be detected, as illustrated in Figure 2 and observed in Equation (10). The 'code' in the time-shifted traffic can synchronize with the one in the original traffic, causing *periodic peaks* in the MSAC, which reveals the *self-similarity* of embedded DHSS watermarks occurring at regular intervals. Intuitively, Theorem 1 provides the information to infer the code length l, as stated in Corollary 1.

Corollary 1. *The code length l is equal to the interval between two consecutive peaks of $E(r^2(\tau))$.*

Proof. From Equation (12), we know that $E(r^2(\tau))$ shows peaks when $\tau = 0$ or $\tau = kl$ (where $k \in [1, w-1]$). Therefore, the $E(r^2(\tau))$ shows peaks with a period of l.

We make the following important observations from Theorem 1 and Corollary 1.

(1) During the derivation of Theorem 1 and Corollary 1, we assume a general type of PN code. This means our theory is applicable to a DHSS-based traceable system using various types of PN codes, even *cryptographically secure* pseudorandom number generators.
(2) $E(r^2(\tau))$ of the PN code-modulated traffic shows peaks of value $\frac{A^4}{w-k}$ when $\tau = kl, 1 \le k < w$. k is the time-shift factor and corresponds to the number of complete bits shifted before calculating the MSAC. The larger the k, the higher the peak value. The highest peaks occur when $\tau = 0$ and $\tau = (w-1)l$.
(3) We can infer the code length l on the basis of the periodicity of $E(r^2(\tau))$.

These distinguishing properties provide features of DHSS watermarks and permit detection. The detection framework will be presented in detail in Section 4.3. We give a simple example to illustrate how to calculate MSAC in Appendix C.

4.2. Mean-square autocorrelation in a nonsynchronized window

In reality, because we do not know the boundary between DHSS watermarks created by an adversary tracing anonymous flows, a traffic segment most likely will not be chosen at the start of a modulated signal bit, nor is its length likely to be a multiple of the code length. This is shown in Figure 3, where the code length is $l = 5$, and we have 14 chunks from the traffic segment, corresponding to 14 chips. In the following, we will consider this case and show that MSAC still shows periodicity. In Figure 3, the traffic segment consists of three chips of modulated bit x_0, two complete modulated bits x_1 and x_2, and one chip of modulated bit x_3. When lag $\tau \neq kl$, the nonsynchronized PN codes in the original traffic segment and its time-shifted version produces a minimal $r(\tau)$ and, thus, a minimal $r^2(\tau)$. When $\tau = kl$ (e.g., $k = 1$, $l = 5$, and $\tau = 5$), as shown in Figure 3, one kind of self-synchronization is revealed, and we derive the peak of $r^2(\tau)$.

Corollary 2 gives an approximate estimation of $E(r^2(\tau))$ for this case of DHSS watermarks in a nonsynchronized window. Its proof can be found in Appendix D.

x_0c_0	x_0c_1	x_0c_2	x_0c_3	x_0c_4	x_1c_0	x_1c_1	x_1c_2	x_1c_3	x_1c_4						
					x_0c_0	x_0c_1	x_0c_2	x_0c_3	x_0c_4	x_1c_0	x_1c_1	x_1c_2	x_1c_3	x_1c_4

Figure 2. Self-similarity of the pseudo-noise code-modulated traffic (first row, a traffic segment; second row, the time-shifted version of the segment).

x_0c_2	x_0c_3	x_0c_4	x_1c_0	x_1c_1	x_1c_2	x_1c_3	x_1c_4	x_2c_0	x_2c_1	x_2c_2	x_2c_3	x_2c_4	x_3c_0					
					x_0c_2	x_0c_3	x_0c_4	x_1c_0	x_1c_1	x_1c_2	x_1c_3	x_1c_4	x_2c_0	x_2c_1	x_2c_2	x_2c_3	x_2c_4	x_3c_0

Figure 3. A nonsynchronized window (first row, a traffic segment; second row, the time-shifted version of the segment).

Corollary 2. *In an experiment, traffic segments containing w bits appear with a probability of p, whereas traffic segments containing w-1 bits appear with a probability of q*, where $q = 1 - p$.

$$E\left(r^2(\tau)\right) \approx \begin{cases} A^4, & \tau = 0, \\ p\dfrac{A^4}{w-k} + q\dfrac{A^4}{w-1-k}, & \tau = kl, 0 < k < w-1, \\ pA^4, & \tau = kl, k = w-1, \\ 0, & \tau \neq kl, 0 \leq k < w \end{cases} \tag{13}$$

where k is an integer.

We have a few observations from Corollary 2:

(1) The MSAC of DHSS watermarks in a nonsynchronized window still demonstrates periodicity. The code length l is equal to the interval between two consecutive peaks of the MSAC.
(2) The peaks of $E(r^2(\tau))$ at the largest lag, $\tau = (w-1)l$, may not have the maximum value as in Theorem 1.

4.3. Framework of detecting DHSS watermarks

In Section 4.1, we demonstrated that the MSAC of DHSS watermarks shows periodicity and may be used in detecting malicious DHSS-based traceable. Such detection does not require any traffic synchronization. In this section we, present a framework using this feature to detect DHSS watermarks.

Figure 4 shows the four stages in detecting DHSS watermarks via the MSAC.

(1) *Acquire traffic*. Target traffic is intercepted by sniffing software such as *tcpdump*. The traffic is divided into segments of equal duration. Each traffic segment is divided into contiguous chunks of t_a seconds (the sampling period), and the average traffic rate for each chunk is calculated, providing a sampled traffic rate time series. Because the detection scheme is based on the autocorrelation of multiple bits DHSS watermarks, the acquired traffic segments must be longer than one signal bit. On the basis of the Nyquist–Shannon sampling theorem [22], an appropriate segment must be sampled with a period of t_a smaller than half the chip duration t_c. We assume that t_a is small enough and t_c is a multiple of t_a (where $t_a = t_c/N$, $N \geq 2$ is an integer) for ease of analysis. In practice, heuristic approaches can be used in determining t_a. In our experiments, 0.1 s is a good selection for t_a. In the practical blind detection of DHSS watermarks, the unit of τ in Equation (8) is t_a.
(2) *Filter direct component*. The traffic rate time series of a traffic segment is then passed into a high-pass filter. The purpose of this process is to remove the direct component because our analysis in Section 4.1 is for data in the bipolar format. We can use the *fast Fourier transform* (FFT) to achieve this: (i) calculate the FFT of the time series, (ii) change the frequency component at zero frequency to *zero* in order to remove the direct component, and (iii) use the reverse FFT to derive data without the direct component.
(3) *Calculate mean-square autocorrelation*. For each segment of the transformed data from Step 2, we compute its square autocorrelation. To estimate the MSAC, we need a few traffic segments and then apply Equation (16)

$$E\left(r^2(\tau)\right) \approx \frac{1}{M}\sum_{i=1}^{M} r_i^2(\tau) \tag{14}$$

where M is the number of segments and $r_i^2(\tau)$ is the MSAC at lag τ for the ith traffic segment.

(4) *Classify by decision rule*. When the MSAC, E $(r^2(\tau))$, is derived, an appropriate decision rule is applied to determine whether the traffic is watermarked. An intuitive decision rule is based on Theorem 1: *if the MSAC demonstrates periodicity, the traffic is DHSS-watermarked*. Note that periodicity means that peaks of the MSAC appear at regularly spaced intervals.

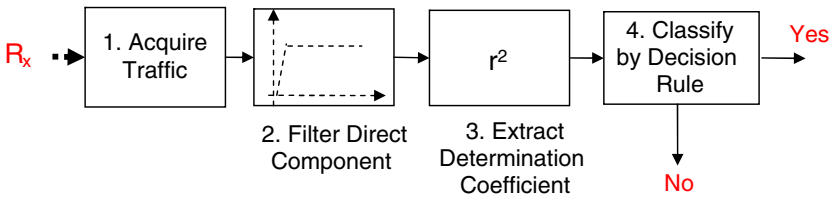

Figure 4. Workflow in detecting direct sequence spread spectrum watermarks.

4.4. Decision rule

We have discussed computing the MSAC and the steps of acquiring traffic and filtering the direct component. We now focus on the decision rule in detecting DHSS watermarks. Theorem 1 implies an intuitive decision rule: *if peaks appear at regular intervals, the traffic is DHSS-watermarked*. Visualization of the MSAC, in terms of lag τ, will clearly demonstrate the signal bit duration. The effectiveness of the decision rule depends on identifying periodic peaks in the MSAC, $E(r^2(\tau))$. If $E(r^2(kl)) > E(r^2(\tau))$, where $\tau \neq kl$, peaks will appear. Thus, $E(r^2(\tau))(\tau \neq kl)$ can be viewed as 'noise', and $E(r^2(kl))$ can be treated as 'signal'. If the signal-to-noise ratio (SNR) is large enough, peaks become apparent. In the following, we first calculate the SNR and then suggest a design of effective decision rules.

The following results are for the synchronized window case. Similar results can be obtained for the nonsynchronized window case. The realistic model of a traffic sample considers both signal components and noise components,

$$y_i = x_i + \xi_i \tag{15}$$

where x_i is the random variable of the modulated signal and ξ_i the random variable of noise. We can then calculate the mixed signal y_i's autocorrelation. This leads to Theorem 2 for the noise MSAC. Its proof can be found in Appendix E.

Theorem 2. *The noise MSAC can be estimated as follows*

$$E\left(r_\xi^2(\tau)\right) = \begin{cases} \delta^4\left(\dfrac{1}{wl}+1\right), & \text{if } \tau = 0, (18) \\ \dfrac{\delta^4}{wl-\tau}, & \text{if } \tau \neq 0 \end{cases} \tag{16}$$

where δ^2 is the noise variance.

Considering Theorems 1 and 2, we have Theorem 3 for the signal-to-noise ratio.

Theorem 3. *The SNR for the MSAC can be estimated as follows*

$$\frac{E\left(r_x^2(\tau)\right)}{E\left(r_\xi^2(\tau)\right)} = \begin{cases} \dfrac{A^4}{\delta^4}/\left(\dfrac{1}{wl}+1\right), & \text{if } \tau = 0, (20) \\ l\dfrac{A^4}{\delta^4}, & \text{if } \tau = kl, 1 \le k < w, (21) \\ 0, & \text{if } \tau \neq kl, 1 \le k < w \end{cases} \tag{17}$$

Proof. On the basis of $E\left(r_x^2(\tau)\right)$ in Equation (12) of Theorem 1 and $E\left(r_\xi^2(\tau)\right)$ in Equation (18) of Theorem 2, the SNR for the MSAC can be derived via straightforward algebraic substitutions.

From Theorem 3, we can see that the SNR of the MSAC shows peaks of $l\frac{A^4}{\delta^4}$ when $\tau = kl$, where $0 < k < w$. Moreover, the longer the code length l, the higher the peak. Therefore, a longer code makes the recognition of DHSS watermarks easier.

We introduce an automatic rule in Algorithm 23 in detecting DHSS watermarks via periodicity of peaks in $E(r^2(\tau))$. We use a heuristic approach. We first guess a code length, l, and then choose parameters for the number of bits in one window, w, and the number of windows (traffic segments), M. From the traffic samples, we then derive the distribution of the MSAC of noise and the mean MSAC of the signal at positions corresponding to integer multiples of the signal bit. Finally, given a false positive rate, we use hypothesis testing to make a decision. If the decision is negative (no DHSS watermarks), we guess another bit length and continue as before, until we have found watermarks or we have exhausted all the bit length choices from a predefined pool.

We can define the *detection rate, P_D*, as the probability that DHSS watermarks are detected, and the *false positive rate*, P_F, as the probability that traffic without DHSS watermarks is misclassified as traffic modulated by a PN code. We can adjust η in Algorithm 23 and derive the corresponding detection rate and false positive rate for each case. It will be shown that this algorithm is effective because the detection rate can be high, whereas the false positive rate is kept small.

4.5. Parameters affecting the blind detection of spread spectrum flow watermarks

We now discuss the impact of the PN code length and the number of signal bits on the blind detection of spread spectrum flow watermarks. On the basis of the results shown in Theorem 3, we know that the SNR increases linearly with the code length.

Algorithm 1: Detecting DHSS Watermarks

Require: (a) l, a value chosen from a (finite) pool of hypothetical code lengths; (b) t_a, a sampling period (so the bit duration would be $l \times t_a$ seconds); (c) w, the number of complete bits per window, setting the window size; (d) M, the predefined number of traffic segments analyzed; and (e) η, a predefined factor controlling the false positive rate (If we assume the noise is Gaussian white noise and $\eta = 3$, the false positive rate is below 2%.).

Ensure: The result is *true* if the traffic is PN code-modulated.

1: While untested code lengths remain do
2: Select an untested value for code length l
3: Determine reasonable values for t_a and w, based on l

```
4: Calculate μ_ξ, the mean of noise MSAC; δ_ξ, standard
   deviation of r_ξ^2(τ), where τ ≠ klt_a and τ < (w − 1) lt_a
5: Calculate the estimated signal MSAC r^2_{x,i} = 1/(w−1)
   Σ_{k=1}^{w−1} r_x^2(klt_a) for the ith traffic segment
6: if 1/M Σ_{i=1}^{M} r^2_{x,i} > μ_ξ + ηδ_ξ then
7:    return result ⇐ true
8:  end if
9: end while
10: return result ⇐ false
```

l. The longer the code length, the higher the SNR and the easier the detection of DHSS watermarks. This raises a question: can attackers use a shorter code length to make DHSS traceable harder to detect? Unfortunately, this is a dilemma for attackers: as seen in Lemma 1 of [7], achieving reasonable traceable accuracy requires a relatively *longer* code length.

Recall that because our detection relies on the autocorrelation of DHSS watermarks, the DHSS watermarks must contain multiple bits (≥ 2). Another question is: can attackers use just one bit to conduct the traceable to escape detection? The answer is no. When there is just a single bit, the false positive rate of the traceable will be too significant at 50% [7]. Given a target false positive rate of 1%, the number of bits must be larger than seven. This will favor the detection of DHSS watermarks again. We will show in Sections 5 and 5.6 that seven bits or fewer is enough in detecting DHSS watermarks.

4.6. Reconstruction of DHSS code

As we can see, by using the approach discussed earlier, the existence of DHSS watermarks can be recognized. The next question becomes: can the DHSS code be successfully reconstructed? If so, the sender and receiver may introduce marks into other traffic and mislead the malicious traceable. We now discuss how to reconstruct the DHSS code effectively.

Similar to the blind detection of DHSS watermarks, we first obtain a segment of the traffic, which is modulated by the PN code. We then remove the direct component using a high-pass filter discussed in Section 4.3.

To simplify our analysis, we assume that the segment of traffic, y lasting for T_s, contains l_0 chips of the end of a signal bit x_k and $l - l_0$ chips of the start of the signal bit x_{k+1}. In practice, Corollary 1 gives T_s, which is the interval between two peaks of the MSAC. Because the MSAC of the filtered traffic shows a peak at the lag where the filtered traffic and its shifted version are synchronized in terms of the PN code, we can determine y from the lag that produces the peak. Determining where y starts can be hard if the traffic contains heavy noise. In such a case, we may not be able to recover the PN code. Therefore, if the assumption holds, we have

$$y = x_k C_e + x_{k+1} C_s + \vec{\xi} \tag{18}$$

where vectors C_e and C_s have a length of l, $\rightarrow \xi$ is the noise random variable, and

$$C_e = \{\text{the last } l_0 \text{ chips of } {}'x_k, \text{ and } l - l_0 \text{ 0s}\} \tag{19}$$

$$C_s = \{l_0 \text{ 0s, and the first } l - l_0 \text{ chips of } x_{k+1}\}' \tag{20}$$

where symbol $'$ refers to the matrix transpose. Therefore, C_e and C_s are column vectors.

Then, we can derive the correlation matrix $\mathbf{R_y}$ of y.

$$\mathbf{R_y} = E((x_k C_e + x_{k+1} C_s + \vec{\xi}) * (x_k C_e + x_{k+1} C_s + \vec{\xi})') \tag{21}$$

$$\begin{aligned} &= E(x_k^2 C_e * C_e' + x_k x_{k+1} C_e * C_s' + x_k C_e * \vec{\xi}' \\ &\quad + x_{k+1} x_k C_s * C_e' + x_{k+1}^2 C_s * C_s' + x_{k+1} C_s * \vec{\xi}' \\ &\quad + x_k \vec{\xi} * C_e' + x_{k+1} \vec{\xi} * C_s' + \rightarrow \xi * \vec{\xi}') \end{aligned} \tag{22}$$

$$= E(x_k^2) C_e * C_e' + E(x_{k+1}^2) C_s * C_s' + E(\vec{\xi} * \vec{\xi}') \tag{23}$$

$$= A^2 C_e * C_e' + A^2 C_s * C_s' + \sigma^2 \mathbf{I} \tag{24}$$

where $\mathbf{I}$ is the identity matrix.

In the following, we try to derive the eigenvalues and eigenvectors of R_y. From the construction of C_e and C_s, we can derive

$$\begin{aligned} \mathbf{R_y} C_e &= (A^2 C_e * C_e' + A^2 C_s * C_s' + \sigma^2 \mathbf{I}) C_e, \\ &= A^2 C_e * C_e' * C_e + A^2 C_s * C_s' \end{aligned} * C_e \tag{25}$$

$$+ \sigma^2 \mathbf{I} * C_e \tag{26}$$

$$= A^2 C_e l_0 + 0 + \sigma^2 C_e \tag{27}$$

$$= (A^2 l_0 + \sigma^2) C_e \tag{28}$$

$$= \sigma^2 \left(1 + \frac{A^2}{\sigma^2} l_0\right) C_e \tag{29}$$

Denote $\rho = \frac{A^2}{\sigma^2}$ as the SNR, and we have

$$\mathbf{R_y} C_e = \lambda_e C_e = \sigma^2 (1 + \rho l_0) C_e \tag{30}$$

Hence, one eigenvector of $\mathbf{R_y}$ is C_e, and the corresponding eigenvalue is λ_e, as follows

$$\lambda_e = \sigma^2 (1 + \rho l_0) \tag{31}$$

Using a similar approach, we can also derive a second eigenvector C_s, and its eigenvalue is λ_s, as follows

$$\lambda_s = \sigma^2 (1 + \rho (l - l_0)) \tag{32}$$

In order to derive all the eigenvalues λ of $\mathbf{R_y}$ in Equation (29), we need to solve the following equation

$$det(\mathbf{R_y} - \lambda \mathbf{I}) = 0 \quad (33)$$

where *det*(.) refers to the calculation of the matrix determinant.

Theorem 4. *The determinant of* $\boldsymbol{R_y} - \lambda\, \boldsymbol{I}$ *is given in Equation (39),*

$$det(\mathbf{R_y} - \lambda \mathbf{I}) = (-1)^l(\lambda - \lambda_e)(\lambda - \lambda_s)(\lambda - \sigma^2)^{l-2} \quad (34)$$

Therefore, $\mathbf{R_y}$ has three eigenvalues: λ_e, λ_s, and σ^2, where $\lambda_e, \lambda_s \geq \sigma^2$. The two eigenvectors λ_e and λ_s, corresponding to the two biggest eigenvalues R_e and R_s, compose the DHSS code as shown in Equations (24) and(25).

The proof of Theorem 4 is given in Appendix F. From Theorem 4, we can derive the DHSS code from eigenvectors C_e in Equation (24) and C_s in Equation (25), corresponding to the two principal eigenvalues λ_e and λ_s. Because $\mathbf{R_y}$ is a symmetric real matrix, we can diagonalize it into the following matrix Λ,

$$\Lambda = \begin{pmatrix} \sigma^2(1+\rho l_0) & 0 & 0 & \cdots & 0 \\ 0 & \sigma^2(1+\rho(l-l_0)) & 0 & \cdots & 0 \\ 0 & 0 & \sigma^2 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & \sigma^2 \end{pmatrix} \quad (35)$$

Using the results obtained above, we can apply the following steps to derive the original DHSS code practically: (i) deriving the correlation matrix $\mathbf{R_y}$; (ii) solving $\mathbf{R_y}$ for its eigenvalues and eigenvectors; and (iii) deriving the DHSS code: the two eigenvectors R_e and R_s, corresponding to the two biggest eigenvalues, compose the DHSS code as stated in Theorem 4. R_s's first few elements will be very small and treated as zeros. Its other elements are treated as 0 s (if the element < 0) or 1 s (if the element > 0) and compose the first part of the PN code. Similarly, the nontrivial elements of R_e compose the remaining part of the PN code. In Section 5.4, we will present simulation results and validate the feasibility of this approach in reconstructing the DHSS code.

5. PERFORMANCE EVALUATION

We have theoretically studied the detection of malicious DHSS-based traceable in previous sections. In this section, we use ns-2 simulations to validate our theory of blind detection of DHSS watermarks. The disadvantage of the multiflow detection approach in [8] is presented in Section 5.5. Results of empirical tests over Tor will be presented in Section 5.6. We have conducted a large number of simulations and all of them corroborate our previous theoretical analysis. In addition, we have developed prototype tools and conducted real-world experiments over *Tor* [10], a popular anonymous communication system to validate our findings. Figure 13 summarizes the trend of the detection rate and the false positive rate in terms of the number of segments, window size, PN code length, and SNR.

5.1. Simulation setup

Figure 5 shows the simulation topology. In Figure 5, n_5 and n_7 are Tor-like mixes [9] (no batching or reordering because they are not practical [23]), as used in teYuFu: DSSS: 2007. The target File Transfer Protocol (FTP) flow runs from node n_0 to node n_8 throughout the simulations. There are also cross flows as noise for the duration of each simulation. In our simulation, the *interferer* uses User Datagram Protocol (UDP) constant bit rate (CBR) traffic to modulate the target FTP flow. The CBR traffic runs from n_1 to n_4 and is an on-off traffic source sharing the link between n_2 and n_3 with the target FTP flow. As we know from the TCP flow control, when the CBR traffic rate increases, the FTP traffic rate decreases; whereas when the CBR traffic rate decreases (e.g., no CBR traffic), the FTP traffic rate increases.

In our simulation, the CBR interference traffic is turned off when a chip within a signal modulated by the PN code is + 1, and it is turned on when the chip is −1. The on-interval and off-interval are equal to the chip duration. In this way, the malicious interferer can mark the interested FTP flow by adjusting its rate through the interference of the CBR traffic.

5.2. Evaluation metric

We use the detection rate P_D and the false positive rate P_F as our evaluation metrics in detecting DHSS watermarks. Recall that we define the *detection rate* P_D as the probability that traffic modulated by PN code is detected as watermarked. The *false positive rate*, P_F, is the probability that unmarked traffic is misclassified as watermarked. The detection rate and false positive rate are illustrated in Figure 6, where $f_0(x)$ is the noise MSAC distribution, $f_1(x)$ the signal *MSAC* distribution, and γ is determined by η in Algorithm 23. We can see that detection rate P_D and false positive rate P_F have an interesting relationship. Both P_D and P_F decrease to zero as γ increases, whereas both P_D and P_F increase to one as γ decreases. A common means of displaying the relationship between P_D and P_F is with a *receiver operating characteristic* (*ROC*) curve, which is a plot of P_D versus P_F. When we try to detect traffic containing malicious

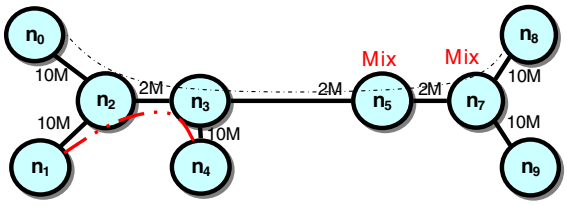

Figure 5. Topology in ns-2.

DHSS watermarks, we want a high detection rate and a low false positive rate.

5.3. Detecting DHSS marks

Now, we show the MSAC-based approach is effective in detecting DHSS watermarks. Figure 7 uses Equation (16) to estimate the MSAC. In this case, the PN code length is seven. Only two traffic segments are used, and each segment contains about three bits. So totally, at most, six bits are used in this nonsynchronized case. The chip duration t_c is 0.5 s, and the sampling interval is 0.1 s. The interference traffic of the CBR traffic rate is 1.2 Mbps. We have a couple of observations in Figure 7. First, peaks indeed appear at multiples of the bit period. The bit length is $7 \times 0.5 = 3.5$ s. Second, false positives may occur because there are some peaks at unexpected places.

Figure 8 shows the detection rate and false positive rate in terms of the number of segments. Other parameters are the same as the previous paragraph. Figure 9 shows the ROC curve when the number of segments is four. We have the following observations in Figures 8 and 9. First, the detection rate approaches 100%, and the false positive rate approaches 0% as the number of segments increases. This validates our theoretical analysis in Section 4. Because there are peaks at the predicted locations, by aggregating enough samples (M, the number of segments), we can recognize these peaks with high certainty. Second, we can achieve a high detection rate while maintaining a low false positive rate. In Figure 8, the false positive rate is always below 10%, whereas the detection rate is more than 60%. The ROC curve in Figure 9 increases sharply when the false positive rate is lower than 10%.

Figure 10 shows the detection rate and false positive rate in terms of the window size (the size of a segment). In this case, the number of segments is set to two. We can see that when the window size increases, the detection rate increases and false positive rate decreases. This demonstrates the feasibility and effectiveness of our decision rule in Section 4.4. As the window size increases, there will be more possible peaks at expected positions. Because our decision rule considers the effect of all those peaks, a better detection performance is as expected.

Figure 11 shows the detection rate and false positive rate in terms of the PN code length. In this case, the

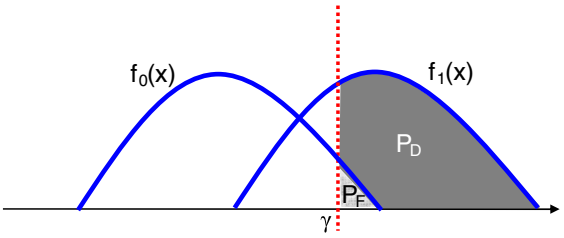

Figure 6. Calculation of P_D and P_F.

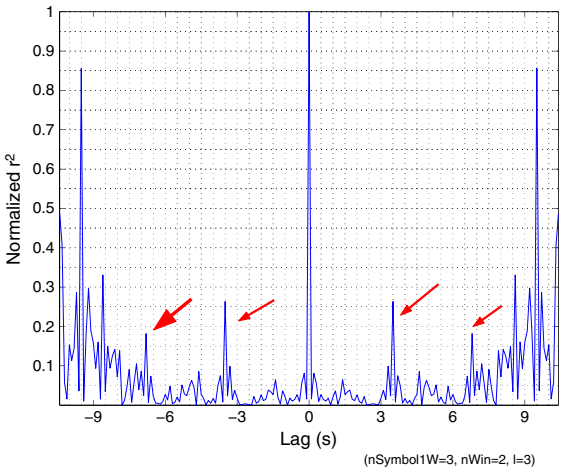

Figure 7. Estimation of mean-square autocorrelation.

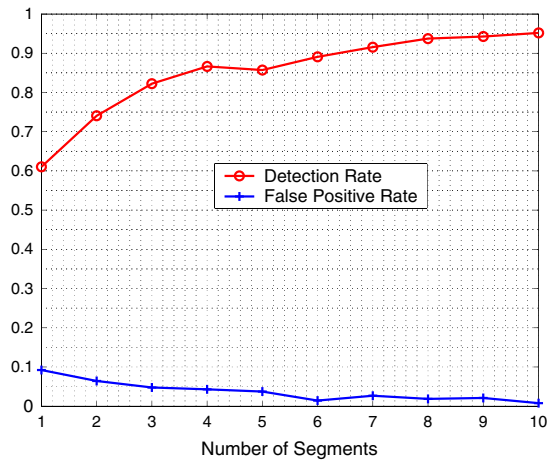

Figure 8. Detection rate and false positive rate versus number of segments.

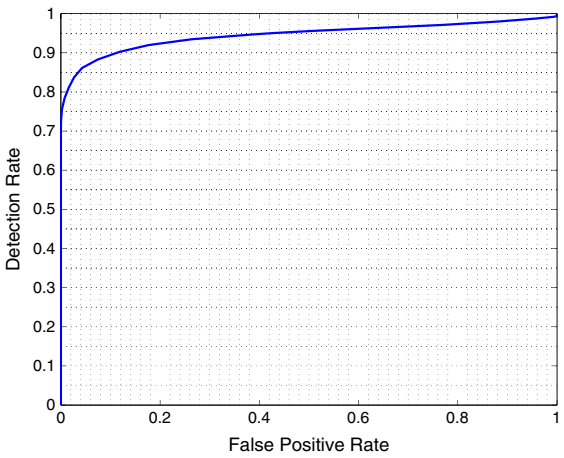

Figure 9. The receiver operating characteristic.

Security Comm. Networks 2013; **6**:257–274 © 2012 John Wiley & Sons, Ltd.
DOI: 10.1002/sec

number of segments is two and window size is three. On the basis of Theorem 3, we know when the PN code length increases, the SNR increases. This will dramatically increase the detection rate and reduce the false positive rate, as shown in Figure 11.

Figure 12 shows the detection rate and false positive rate in terms of interference intensity. The PN code length is seven, the number of segments is four, and window size is three. When the interference intensity increases, SNR increases generally. A larger SNR will reduce the false positive rate and increase the detection rate, as shown in Figure 12.

Figure 13 summarizes the trend of the detection rate and false positive rate in terms of the number of segments, window size, PN code length, and SNR. The symbol ↗ refers to the increasing function. For example, the detection rate is an increasing function of the number of segments. The symbol ↘ refers to the decreasing function. For example, the false positive rate is a decreasing function of the number of segments.

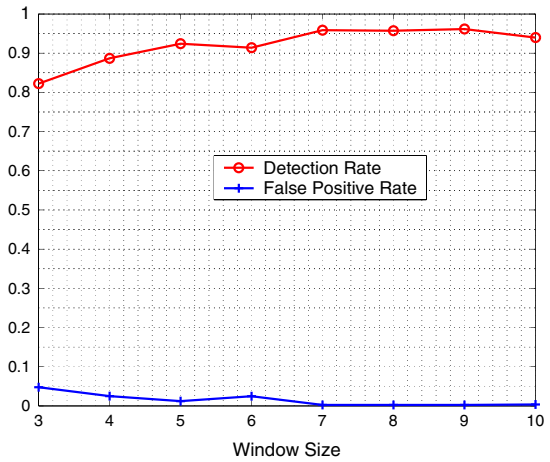

Figure 10. Detection rate and false positive rate versus window size.

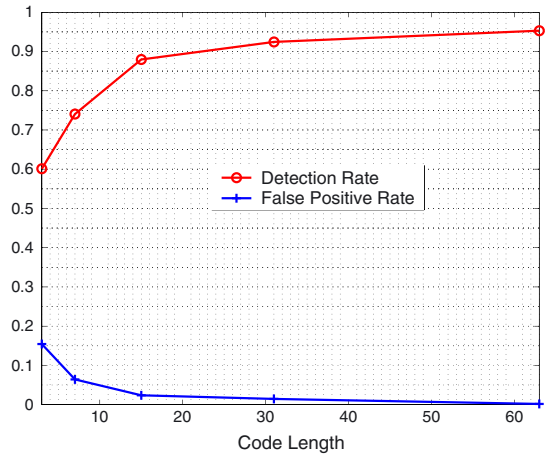

Figure 11. Detection rate and false positive rate versus pseudo-noise code length.

5.4. Reconstruction of DHSS code

In Section 4.6, we discussed the reconstruction of the DHSS code in case of Gaussian noise and random signal. We conducted simulations within Matlab (Natick, Massachusetts, U.S.A.) and validated the feasibility of the theory. Figure 14 illustrates the recovery ratio versus the SNR. The DHSS code is seven bits long. We make the following observations: (i) with the increasing SNR, we can recover more and more bits correctly. When SNR approaches 1, the recovery ratio reaches 100% when a 15-bit signal is sent. Even with $SNR = 0.36$, the recovery ratio for a 15-bit signal is around 80%. If an attacker knows the type of DHSS code such as the m-sequence code, they may utilize the properties of such a code and further increase the recovery ratio. (ii) With the increasing number of bits in a signal, the recovery ratio increases. This is because more signal bits render a more accurate correlation matrix, given in Equation (29). A more accurate correlation matrix produces more accurate eigenvectors that compose the DHSS code by utilizing Theorem 4.

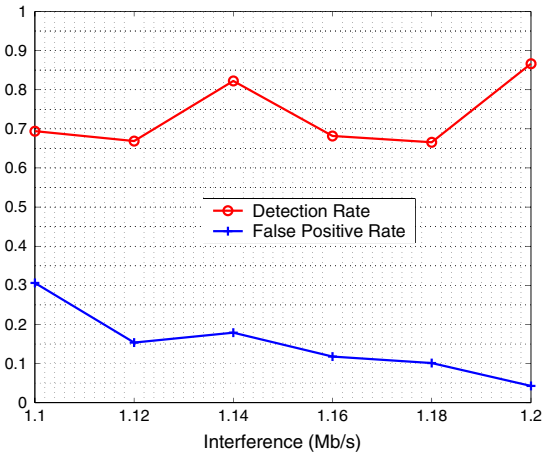

Figure 12. Detection rate and false positive rate versus signal-to-noise ratio.

	Detection rate	False positive rate
Number of segments	↗	↘
Window size	↗	↘
PN code length	↗	↘
Signal to noise (SNR) ratio	↗	↘

Figure 13. Summary of detection rate and false positive rate trend.

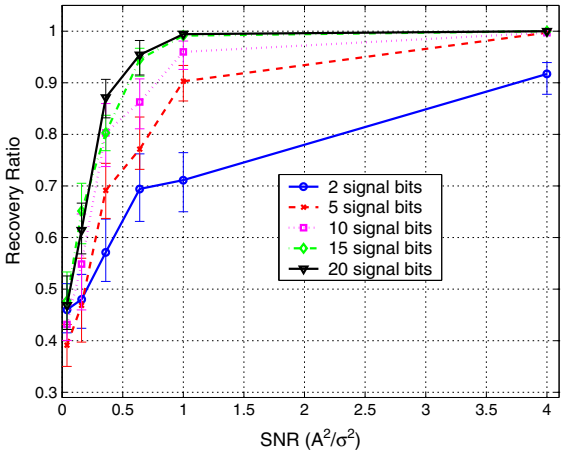

Figure 14. Reconstruction of the pseudo-noise code.

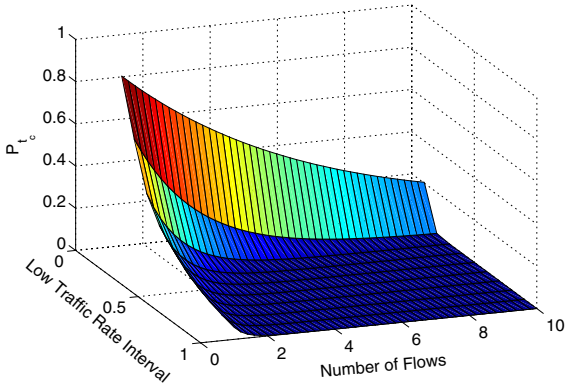

Figure 15. Multiflow detection [8] of the direct sequence spread spectrum-based traceable.

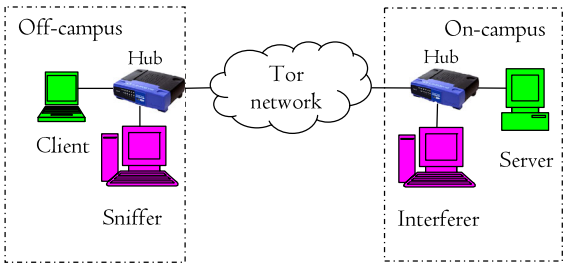

Figure 16. The experiment setup.

5.5. Deficiency of multiflow detection in [8]

In [8], a multiflow approach is proposed to detect interval-based watermarks [19,2] and DHSS-based watermarks [7]. In the case of detecting DHSS watermarks, they use the MMPP to model the lasting interval of a low traffic rate. Denote the probability that a low traffic rate lasts for more than a chip duration t_c as P_{t_c}. When the number of flows increases and P_{t_c} is smaller than a threshold, they can decide whether the traffic is watermarked.

On the basis of Equation (6) in [8], we draw P_{t_c} in terms of the number of flows and interval of low traffic rate in Figure 15. The MMPP model used in Figure 15 is trained using FTP downloading sessions over Tor. We can see that the multiflow detection approach cannot detect a DHSS watermarked flow when the chip duration t_c varies from 0.1 to 1.0 s, and there is only one watermarked flow, given a threshold of 1%. When $t_c = 0.2$ s, eight flows are required to detect watermarked flows. When $t_c = 0.4$ s, four flows are required. Requiring multiple simultaneous watermarked flows definitely limits the applications of this approach. It is not always easy to find multiple simultaneous watermarked flows. However, the advantage of the approach in [8] is that they can detect interval-based watermarks [19,2] given a sufficient number of flows.

5.6. Experiments over Tor

To validate our findings, we developed prototype tools and conducted real-world experiments over *Tor* [10], a popular anonymous communication system. Figure 16 shows the experimental setup, which represents a typical use of *Tor* for anonymous file transfer or web browsing. All machines are configured with Fedora Core 3 (http://fedoraproject.org/). We downloaded a file from a web server on a university campus to an off-campus computer as a client. The downloading software was *wget* (http://www.gnu.org/software/wget/) with the appropriate proxy configuration in order to use Tor.

In order to carry out a traceback, we set up two more computers. One computer, used as an *interferer*, sends an appropriate volume of traffic to the server. Another computer is used as a *sniffer* to collect the traffic destined for the client computer. The *interferer* and server were connected by a hub, as were the *sniffer* and the client computer. We use this simple approach to investigate the detection of the malicious DSSS-based traceback, even though the interference is not optimal. There are more efficient approaches for interference (such as dropping packets at a malicious Tor router).

Figure 17 shows one case of detecting DSSS watermarks over Tor, where the upper chart shows the traffic rate varying with time. The lower chart shows the emerging periodicity on the basis of calculating the MSAC of traffic segments from the data set in the upper chart. The setup for this DSSS watermark detection test was the chip duration, $t_c = 3$ s, and the code length, $l = 7$. So, the bit duration is 21 s, as used in [7]. In Figure 17, we can see that the MSAC indeed demonstrates periodicity at positions by multiple of 21 s, and we can effectively detect DSSS watermarks.

6. CONCLUSION

In this paper, we proposed an approach for blindly detecting malicious DSSS traceback, which may be applied to

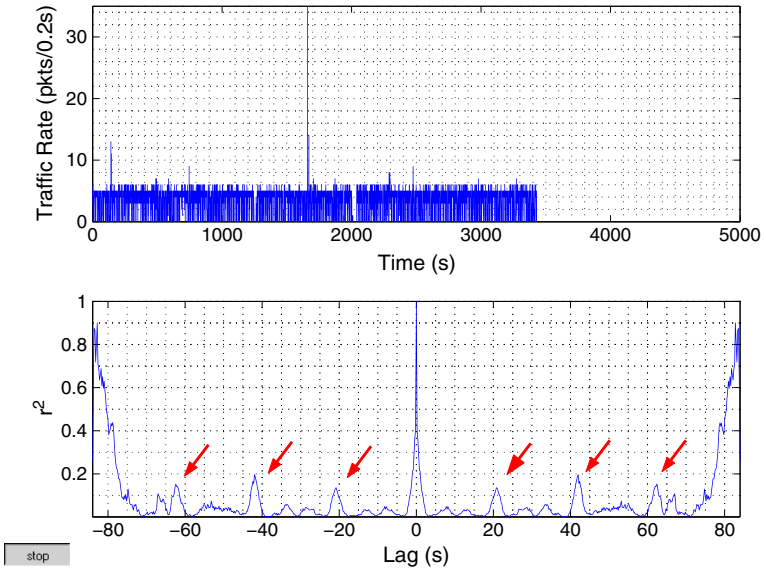

Figure 17. Detecting direct sequence spread spectrum marks over Tor.

trace anonymous traffic flows and seriously degrade the anonymity that an anonymous communication network provides. By calculating the MSAC of the DSSS-modulated traffic, we found that the self-similarity introduced by the DSSS watermarks causes periodic peaks visible in the MSAC in terms of the lag. Our detection approach does not require any knowledge of the PN code used by the DSSS watermarking, and there is no need of traffic synchronization. Our results from ns-2 simulations demonstrate a high detection rate with a low false positive rate, and experiments on the real world anonymous communication network *Tor* also verified the feasibility of the attack. Once the DSSS code length is derived, we developed a theory to recover the DSSS code, and our simulations verified the recovery theory.

ACKNOWLEDGEMENTS

This work is partially supported by SAR Hong Kong RGC Competitive Earmarked Research Grant (CERG) No. (CityU 114908) and CityU Applied R & D Funding (ARD) No. 9678002. This work is also supported in part by the US National Science Foundation (NSF) under grants 0943479, 0907964, and CNS-1117175, by CAREER Award CCF 0546668, by the Army Research Office (ARO) under grant No. AMSRD-ACC-R50521-CI. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of those sponsors.

APPENDIX A

Figure 18 shows the basic principle of DSSS. The original signal x at the transmitter is a series of bits ($+1$ or -1). The *bit duration* for both bit $+1$ and -1 is T_s seconds. A PN code C of a series of *chips* $+1$ and -1 is generated at the transmitter and shared between the receivers. Each chip in the PN code lasts for t_c seconds (denoted as *chip duration*), so the chip rate is $f_c = 1/t_c$. l is the number of chips per signal bit and is also called as the *PN code length*. These concepts are illustrated in Figure 19.

Now, we discuss the spreading process at the transmitter, as illustrated in Figure 19. Without loss of generality, we discuss, using a PN code to spread one signal bit, $+1$ or -1. x is directly multiplied with the PN code C, which is independent of the signal, to produce the transmitted signal $X = C\ x$, where C is a $1 \times l$ vector with elements corresponding to the chip values, either $+1$ or -1 drawn from the PN code at the transmitter.

The transmitted signal X passes through the communication channel and reaches the receiver. If there is no interference along the channel, the received baseband signal is X. In order to recover the original signal from X, X is multiplied with the same PN code at the receiver. We have the recovered signal

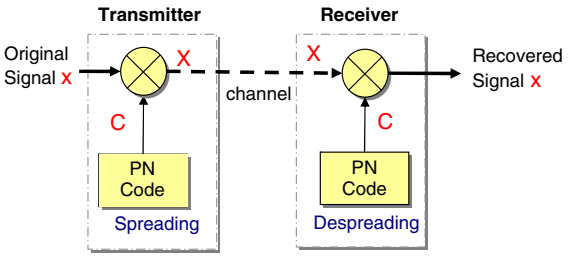

Figure 18. Direct sequence spread spectrum.

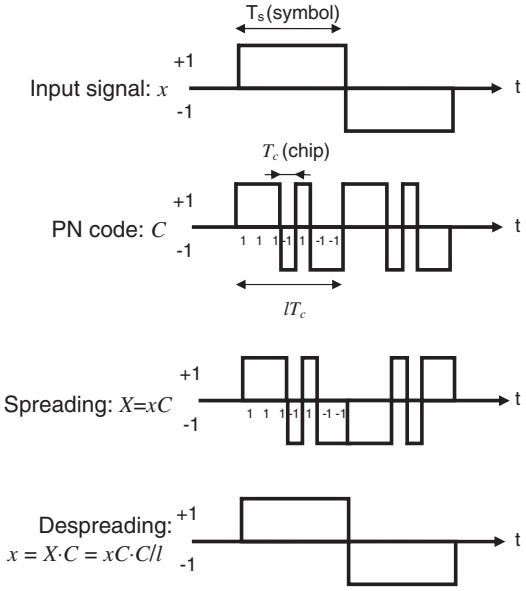

Figure 19. Spreading and despreading in the DSSS.

$$x = \frac{\sum(X \cdot C)}{l} = x\frac{\sum(C \cdot C)}{l} \tag{36}$$

where the operator of · refers to direct multiplication of vectors and the operator of $\sum$ adds up all the elements of a vector. Therefore, a receiver with the right PN code can recover the original signal. This despreading process is illustrated in Figure 19.

If the receiver or a third party does not have the right PN code, or a wrong PN code C', $\sum(C \cdot C')/l \neq 1$, they cannot reproduce the original signal x.

APPENDIX B

In this appendix, we provide the proof of Theorem 1.

Case 1 $\tau = 0$: From Equations (7) and (8), we have

$$r(0) = \frac{1}{wl}\sum_{i=0}^{w-1}\sum_{j=0}^{l-1} x_i^2 c_j^2 \tag{37}$$

Because $x_i^2 = A^2$ and $c_i^2 = 1$, then

$$r(0) = A^2 \tag{38}$$

and

$$E\left(r^2(0)\right) = A^4 \tag{39}$$

Case 2 $\tau = kl$: From Equations (7) and (8),

$$r(kl) = \frac{1}{wl - kl}\sum_{i=0}^{wl-kl-1} y_i y_{i+kl}, \tag{40}$$

$$= \frac{1}{wl - kl}(x_0 x_{0+k} \rightarrow C \cdot \rightarrow C$$

$$+ \cdots + x_{w-1-k}x_{w-1} \rightarrow C \cdot \rightarrow C) \tag{41}$$

where · refers to *dot product* and $\rightarrow C \cdot \rightarrow C = l$. Therefore,

$$r(kl) = \frac{l}{wl - kl}(x_0 x_k + \cdots + x_{w-1-k}x_{w-1}) \tag{42}$$

$$= \frac{1}{w - k}\sum_{i=0}^{w-1-k} x_i x_{i+k} \tag{43}$$

The mean-square autocorrelation (MSAC) $E(r2(kl))$ can be calculated as follows:

$$E\left(r^2(kl)\right) = E\left[\left(\frac{1}{(w-k)^2}\sum_{i=0}^{w-1-k} x_i x_{i+k}\right)^2\right] \tag{44}$$

$$= \frac{1}{(w-k)^2}E\left[\left(\sum_{i=0}^{w-1-k} x_i x_{i+k}\right)^2\right] \tag{45}$$

Recall that x_i and x_j $(i \neq j)$ are independent. Because $E(x_i) = 0$, $k \neq 0$, then

$$E\left(r^2(kl)\right) = \frac{1}{(w-k)^2}E\left(\left(\sum_{i=0}^{w-1-k} x_i x_{i+k}\right)^2\right) \tag{46}$$

$$= \frac{1}{(w-k)^2}E\left(\sum_{i=0}^{w-1-k}\sum_{j=0}^{w-1-k} x_i x_{i+k} x_j x_{j+k}\right) \tag{47}$$

$$= \frac{1}{(w-k)^2}E\left(\sum_{i=j} x_i x_{i+k} x_j x_{j+k} + \sum_{i\neq j} x_i x_{i+k} x_j x_{j+k}\right) \tag{48}$$

$$= \frac{1}{(w-k)^2}\left(\sum_{i=0}^{w-1-k} E\left(x_i^2 x_{i+k}^2\right) + \sum_{i\neq j} E\left(x_i x_{i+k} x_j x_{j+k}\right)\right) \tag{49}$$

Because of x_i, x_{i+k}, x_j, and $x_{j+\tau}$ (e.g., x_i) will be independent from the other three random variables, and $E(x_i) = 0$, $\sum_{i\neq j} E\left(x_i x_{i+k} x_j x_{j+\tau}\right) = 0$. Therefore,

$$E\left(r^2(kl)\right) = \frac{A^4}{(w-k)^2}(w-k+0) \tag{50}$$

$$= \frac{A^4}{w-k} \tag{51}$$

Case 3 $\tau \neq kl$, $0 \leq k < w$: In this case, the autocorrelation can be calculated as follows

$$r(\tau) = \frac{1}{(wl-\tau)} \sum_{i=0}^{wl-1-\tau} x_{[i/l]} c_{i\%l} x_{[(i+\tau)/l]} c_{(i+\tau)\%l} \tag{52}$$

$$= \frac{1}{(wl-\tau)} \sum_{i=0}^{wl-1-\tau} x_{[i/l]} x_{[(i+\tau)/l]} c_{i\%l} c_{(i+\tau)\%l} \tag{53}$$

Equation (58) contains items corresponding to a PN code times its shifted version: $c_{i\%l}c_{(i+\tau)\%l}$, where it is weighted by the product of two independent bits $x_{[i/l]}x_{[(i+\tau)/l]}$. In an ideal case, a PN code has a noise-like autocorrelation function: when the lag τ is nonzero, the autocorrelation is zero, that is $r_c(\tau) = E(c_i c_{i+\tau}) = 0$. The m-sequence code we use in this paper has the best noise-like autocorrelation function among popular PN codes. Therefore, approximately, we can have

$$E\left(x_i x_j c_i c_{i+\tau}\right) = E\left(x_i x_j\right) E(c_i c_{i+\tau}) = 0 \tag{54}$$

$$r(\tau) = 0, \tau \neq kl, 0 \leq k < w \tag{55}$$

Therefore,

$$E\left(r^2(\tau)\right) = 0, \tau \neq kl, 0 \leq k < w \tag{56}$$

After combining the results of the three cases together, the theorem is proved.

APPENDIX C

In this appendix, we demonstrate one example of calculating $E(r^2(\tau))$ in Theorem 1 for an extreme case, illustrating the self-similarity with greater clarity. Assume that the signal has seven bits, all 1's, $\{1,1,1,1,1,1,1\}$ and the PN code has seven chips $\{1,-1,-1,1,1,1,-1\}$. Therefore, the spread signal X is

$$X = \begin{matrix} 1,-1,-1,1,1,1,-1, (62) \\ 1,-1,-1,1,1,1,-1, (63) \\ 1,-1,-1,1,1,1,-1, (64) \\ 1,-1,-1,1,1,1,-1, (65) \\ 1,-1,-1,1,1,1,-1, (66) \\ 1,-1,-1,1,1,1,-1, (67) \\ 1,-1,-1,1,1,1,-1 \end{matrix} \tag{57}$$

where one row corresponds to one spread bit.

Given the spread signal X, we assume that we have collected two traffic samples, each of which contains two complete spread signal bits. Equation (8), repeated here, estimates the autocorrelation of one sample.

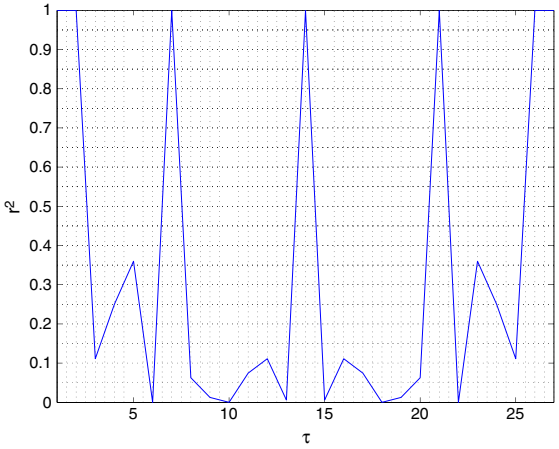

Figure 20. A simple example of calculating $E(r^2(\tau))$.

$$r(\tau) = \frac{1}{(wl-\tau)} \sum_{i=1}^{wl-\tau} y_i y_{i+\tau} \tag{58}$$

Then, we can use two samples to calculate an average for $r^2(\tau)$, that is, an estimation of $E(r^2(\tau))$. Rows 1 and 2 in Equation (62) constitute our first sample, and Rows 3 and 4 constitute our second sample. Through a simple calculation, we can obtain the estimated $E(r^2(\tau))$, as illustrated in Figure 20. We can see that the period of the peaks is indeed seven (the PN code length), as Theorem 1 indicates.

APPENDIX D

In this appendix, we prove Corollary 2.

Proof. Let us derive an approximate estimation for $r^2(\tau)$ in this case. Assume that the window size is $wl + \Delta$, where $\Delta < l$, so that w is the maximum number of complete bits in a window. An actual window may contain a partial bit of s chips at the start, w' complete bits, and a partial bit of e chips at the end of the window.

Therefore,

$$0 < s + e < 2l \tag{59}$$

$$wl + \Delta = s + w'l + e \tag{60}$$

Rearranging Equation (71), we have

$$w'l = wl + \Delta - (s+e) \tag{61}$$

Because $0 < \Delta < l$,

$$-2l < -(s+e) < \Delta - (s+e) < l - (s+e) < l \tag{62}$$

we know

$$wl - 2l < w^{'}l < wl + l \tag{63}$$

So, w' has two possible values: w and w-1.

To estimate the mean of $r^2(\tau)$, we ignore the influence of s and e in each traffic segment. In the case of traffic segments containing w or w-1 bits, we can use Theorem 1 to estimate their MSAC, and $r_w^2(\tau)$ and $r_{w-1}^2(\tau)$, respectively. Then, utilizing the Total Probability Theorem, we know

$$r^2(\tau) = pr_w^2(\tau) + qr_{w-1}^2(\tau) \tag{64}$$

where $r_w^2(\tau)$ and $r_{w-1}^2(\tau)$ can be calculated in Equation (12).

Therefore, assuming k is an integer, we have:

(1) When $\tau = 0$, $r^2(0) = pr_w^2(0) + qr_{w-1}^2(0) = A^4$.
(2) When $\tau = kl$ and $0 < k < w-1$, $r^2(\tau) = pr_w^2(\tau) + qr_{w-1}^2(\tau) = p\frac{A^4}{w-k} + q\frac{A^4}{w-1-k}$.
(3) When $\tau = (w-1)l$, that is $k = w-1$, $r_{w-1}^2(\tau) = 0$, so that $r^2(\tau) = pr_w^2(\tau) + qr_{w-1}^2(\tau) = p\frac{A^4}{w-(w-1)} = pA^4$.
(4) When $\tau \neq kl$ and $0 \leq k < w$, $r^2(\tau) = pr_w^2(\tau) + qr_{w-1}^2(\tau) = p \times 0 + q \times 0 = 0$.

Summarizing the results above, we derive Equation (15) in Corollary 2.

APPENDIX E

In this appendix, we provide the proof of Theorem 2. The realistic model of our traffic sample is a combination of signal components and noise components,

$$y_i = x_i + \xi_i \tag{65}$$

where x_i is the random variable of the modulated signal and ξ_i the random variable of noise.

We can calculate the mixed signal y's autocorrelation as follows,

$$r(y_i y_{i+\tau}) = E((x_i + \xi_i)(x_{i+\tau}\xi_{i+\tau})) \tag{66}$$

$$= E(x_i x_{i+\tau}) + E(x_i \xi_{i+\tau}) + E(\xi_i x_{i+\tau}) + E(\xi_i \xi_{i+\tau}) \tag{67}$$

We assume that the signal and Gaussian white noise are independent and their means are zero. We also assume that the variance of noise is δ^2, Therefore,

$$r(y_i y_{i+\tau}) = E(x_i x_{i+\tau}) + E(\xi_i \xi_{i+\tau}) \tag{68}$$

Apply Theorem 1 to Eqaution (68), we have

$$r(y_i y_{i+\tau}) = \begin{cases} r_x(\tau) + r_\xi(\tau), & \tau = kl, (80) \\ r_\xi(\tau), & \tau \neq kl \end{cases} \tag{69}$$

where $r_x(\tau) = E(x_i x_{i+\tau})$ and $r_\xi(\tau) = E(\xi_i \xi_{i+\tau})$.

Noise time series $\rightarrow \xi$ can be represented as follows,

$$\rightarrow \xi = (\xi_1, \cdots, \xi_{wl}) \tag{70}$$

Then, the correlation of $\rightarrow \xi$ can be estimated as follows,

$$r_\xi(\tau) = \frac{1}{wl - \tau} \sum_{i=0}^{wl-1-\tau} \xi_i \xi_{i+\tau} \tag{71}$$

Therefore, the mean of the noise's MSAC can be calculated as follows,

$$E\left(r_\xi^2(\tau)\right) = E\left(\left(\frac{1}{wl - \tau} \sum_{i=0}^{wl-1-\tau} \xi_i \xi_{i+\tau}\right)^2\right) \tag{72}$$

$$= \frac{1}{(wl - \tau)^2} E\left(\left(\sum_{i=0}^{wl-1-\tau} \xi_i \xi_{i+\tau}\right)^2\right) \tag{73}$$

If $\tau = 0$, we have

$$E\left(r_\xi^2(0)\right) = \frac{1}{(wl)^2} E\left(\left(\sum_{i=0}^{wl-1} \xi_i^2\right)^2\right) \tag{74}$$

$$= \frac{1}{(wl)^2}\left(wl2\delta^4 + (wl)^2\delta^4\right) \tag{75}$$

$$= \delta^4\left(\frac{1}{wl} + 1\right) \tag{76}$$

If $\tau \neq 0$, we have

$$E\left(r_\xi^2(\tau)\right) = \frac{1}{(wl - \tau)^2} \sum_{i=0}^{wl-1-\tau} E\left(\xi_i^2\right) E\left(\xi_{i+\tau}^2\right) \tag{77}$$

$$= \frac{\delta^4}{wl - \tau} \tag{78}$$

APPENDIX F

In this appendix, we prove Theorem 4. Let C_e and C_s have a format as in Equations (91) and (92),

$$C_e = \begin{bmatrix} 1 & -1 & 1 & 0 & 0 \end{bmatrix}^{'} \tag{79}$$

$$C_s = \begin{bmatrix} 0 & 0 & 0 & -1 & 1 \end{bmatrix}^{'} \tag{80}$$

R_y can be written as follows,

$$\mathbf{R_y} = \begin{pmatrix} \mathbf{R_y^U} & 0 \\ 0 & \mathbf{R_y^L} \end{pmatrix} \tag{81}$$

In particular, according to C_e and C_s in Equations (91) and (92), we have

$$\mathbf{R}_{\mathbf{y}}^{\mathbf{U}} = \begin{pmatrix} A^2+\sigma^2 & -A^2 & A^2 \\ -A^2 & A^2+\sigma^2 & -A^2 \\ A^2 & -A^2 & A^2+\sigma^2 \end{pmatrix} \tag{82}$$

and

$$\mathbf{R}_{\mathbf{y}}^{\mathbf{L}} = \begin{pmatrix} A^2+\sigma^2 & -A^2 \\ -A^2 & A^2+\sigma^2 \end{pmatrix} \tag{83}$$

Denote the nonzero items of C_e and C_s as C_{eu} and C_{sl}, and we know

$$\mathbf{R}_{\mathbf{y}}^{\mathbf{U}} = A^2 C_{eu} C_{eu} + \sigma^2 \mathbf{I} \tag{84}$$

$$\mathbf{R}_{\mathbf{y}}^{\mathbf{L}} = A^2 C_{sl} C_{sl} + \sigma^2 \mathbf{I} \tag{85}$$

Therefore, we have

$$det\left(\mathbf{R}_{\mathbf{y}} - \lambda \mathbf{I}\right) = det\left(\mathbf{R}_{\mathbf{y}}^{\mathbf{U}} - \lambda \mathbf{I}\right) \times det\left(\mathbf{R}_{\mathbf{y}}^{\mathbf{L}} - \lambda \mathbf{I}\right) \tag{86}$$

Now, let us calculate $det(\mathbf{R}_{\mathbf{y}}^{\mathbf{U}} - \lambda \mathbf{I})$. In order to do so, we first linearly transform $\mathbf{R}_{\mathbf{y}}^{\mathbf{U}} - \lambda \mathbf{I}$, and we have

$$\mathbf{R}_{\mathbf{y}}^{\mathbf{U}} - \lambda \mathbf{I} = A^2 C_{eu} C_{eu} + \sigma^2 \mathbf{I} - \lambda \mathbf{I} \tag{87}$$

$$= A C_{eu}\left(A C_{eu}\right)' + \left(\sigma^2 - \lambda\right) \rightarrow I \tag{88}$$

Denote $(AC_{eu})'$ as $\rightarrow B = (b_{11}\ \ b_{12} \cdots b_{1l_0})'$. We know b_{1i} is either A or $-A$ and

$$\mathbf{R}_{\mathbf{y}}^{\mathbf{U}} - \lambda \mathbf{I} = \begin{pmatrix} b_{11}B \\ b_{12}B \\ \cdots \\ b_{1l_0}B \end{pmatrix} + \left(\sigma^2 - \lambda\right) \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 \end{pmatrix} \tag{89}$$

We can linearly transform $R_y^U - \lambda I$ as follows,

$$\mathbf{R}_{\mathbf{y}}^{\mathbf{U}} - \lambda \mathbf{I} \rightarrow \begin{pmatrix} 0 \\ 0 \\ \cdots \\ b_{1l_0}B \end{pmatrix} + \left(\sigma^2 - \lambda\right) \begin{pmatrix} 1 & 0 & \cdots & -\dfrac{b_{11}}{b_{1l_0}} \\ 0 & 1 & \cdots & -\dfrac{b_{12}}{b_{1l_0}} \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 \end{pmatrix} \tag{90}$$

$$= \begin{pmatrix} \sigma^2-\lambda & 0 & \cdots & -\dfrac{b_{11}}{b_{1l_0}}\left(\sigma^2-\lambda\right) \\ 0 & \sigma^2-\lambda & \cdots & -\dfrac{b_{12}}{b_{1l_0}}\left(\sigma^2-\lambda\right) \\ \cdots & \cdots & \cdots & \cdots \\ b_{1l_0}b_{11} & b_{1l_0}b_{12} & \cdots & \sigma^2-\lambda+b_{1l_0}^2 \end{pmatrix} \tag{91}$$

We can further transform Equation (91) as follows,

$$\mathbf{R}_{\mathbf{y}}^{\mathbf{U}} - \lambda \mathbf{I} \rightarrow \begin{pmatrix} \sigma^2-\lambda & 0 & \cdots & -\dfrac{b_{11}}{b_{1l_0}}\left(\sigma^2-\lambda\right) \\ 0 & \sigma^2-\lambda & \cdots & -\dfrac{b_{12}}{b_{1l_0}}\left(\sigma^2-\lambda\right) \\ \cdots & \cdots & \cdots & \cdots \\ 0 & b_{1l_0}b_{12} & \cdots & \sigma^2-\lambda+b_{1l_0}^2+b_{11}^2 \end{pmatrix} \tag{92}$$

$$\rightarrow \begin{pmatrix} \sigma^2-\lambda & 0 & \cdots & -\dfrac{b_{11}}{b_{1l_0}}\left(\sigma^2-\lambda\right) \\ 0 & \sigma^2-\lambda & \cdots & -\dfrac{b_{12}}{b_{1l_0}}\left(\sigma^2-\lambda\right) \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \sigma^2-\lambda+b_{1l_0}^2+b_{11}^2+b_{12}^2 \end{pmatrix} \tag{93}$$

$$\rightarrow \begin{pmatrix} \sigma^2-\lambda & 0 & \cdots & -\dfrac{b_{11}}{b_{1l_0}}\left(\sigma^2-\lambda\right) \\ 0 & \sigma^2-\lambda & \cdots & -\dfrac{b_{12}}{b_{1l_0}}\left(\sigma^2-\lambda\right) \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \sigma^2-\lambda+\sum_{i=1}^{l_0} b_{1i}^2 \end{pmatrix} \tag{94}$$

Since b_{1i} is either A or $-A$,

$$\mathbf{R}_{\mathbf{y}}^{\mathbf{U}} - \lambda \mathbf{I} \rightarrow \begin{pmatrix} \sigma^2-\lambda & 0 & \cdots & -\dfrac{b_{11}}{b_{1l_0}}\left(\sigma^2-\lambda\right) \\ 0 & \sigma^2-\lambda & \cdots & -\dfrac{b_{12}}{b_{1l_0}}\left(\sigma^2-\lambda\right) \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \sigma^2-\lambda+l_0A^2 \end{pmatrix} \tag{95}$$

Therefore,

$$det\left(\mathbf{R}_{\mathbf{y}}^{\mathbf{U}} - \lambda \mathbf{I}\right) = (-1)^{l_0}\left(\lambda - \sigma^2\right)^{l_0-1}\left(\lambda - \left(\sigma^2 + l_0A^2\right)\right) \tag{96}$$

Similarly, we can derive

$$det\left(\mathbf{R}_{\mathbf{y}}^{\mathbf{L}} - \lambda \mathbf{I}\right) = (-1)^{l-l_0}(\lambda - \sigma^2)^{l-l_0-1} (\lambda - (\sigma^2 + (l-l_0)A^2)) \tag{97}$$

Finally, we have

$$det(\mathbf{R_y} - \lambda\mathbf{I}) = (-1)^l(\lambda - \sigma^2)^{l-2} (\lambda - (\sigma^2 + l_0A^2)) (\lambda - (\sigma^2 + (l - l_0)A^2)), \quad (98)$$

$$= (-1)^l(\lambda - \sigma^2)^{l-2} (\lambda - \sigma^2(1 + l_0A^2/\sigma^2)) (\lambda - \sigma^2(1 + (l - l_0)A^2/\sigma^2)),$$

$$= (-1)^l(\lambda - \sigma^2)^{l-2} (\lambda - \lambda_e) \quad (99)$$

$$(\lambda - \lambda_s) \quad (100)$$

REFERENCES

1. Ling Z, Luo J, Yu W, Fu X, Xuan D, Jia W. A new cell counter based attack against tor. In *Proceedings of 16th ACM Conference on Computer and Communications Security (CCS)*, 2009.
2. Wang X, Chen S, Jajodia S. Network flow watermarking attack on low-latency anonymous communication systems. In *Proceedings of the 2007 IEEE Symposium on Security and Privacy (S&P)*, May 2007.
3. Hopper N, Vasserman E, Chan-Tin D. How much anonymity does network latency leak?. In *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS)*, 2007.
4. Murdoch SJ, Danezis G. Low-cost traffic analysis of tor. In *Proceedings of the IEEE Security and Privacy Symposium (S&P)*, May 2006.
5. Overlier L, Syverson P. Locating hidden servers. In *Proceedings of the IEEE Security and Privacy Symposium (S&P)*, May 2006.
6. Murdoch SJ. Hot or not: revealing hidden services by their clock skew. In *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS)*, 2006.
7. Yu W, Fu X, Graham S, Xuan D, Zhao W. DSSS-based flow marking technique for invisible traceback. In *Proceedings of the 2007 IEEE Symposium on Security and Privacy (S&P)*, May 2007.
8. Kiyavash N, Houmansadr A, Borisov N. Multi-flow attacks against network flow watermarking schemes. In *Proceedings of USENIX Security*, 2008.
9. Chaum D. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM* 1981; **4**(2).
10. Dingledine R, Mathewson N, Syverson P. Tor: the second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, August 2004.
11. Danezis G, Dingledine R, Mathewson N. Mixminion: design of a type III anonymous remailer protocol. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy (S&P)*, May 2003.
12. Zhu Y, Fu X, Graham B, Bettati R, Zhao W. On flow correlation attacks and countermeasures in mix networks. In *Proceedings of Workshop on Privacy Enhancing Technologies (PET)*, May 2004.
13. Levine BN, Reiter MK, Wang C, Wright M. Timing attacks in low-latency mix-based systems. In *Proceedings of Financial Cryptography (FC)*, February 2004.
14. Fu X, Zhu Y, Graham B, Bettati R, Zhao W. On flow marking attacks in wireless anonymous communication networks. In *Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS)*, April 2005.
15. Wang X, Reeves DS, Wu SF, Yuill J. Sleepy watermark tracing: an active network-based intrusion response framework. In *Proceedings of 16th International Conference on Information Security (IFIP/Sec)*, June 2001.
16. Wang X, Reeves DS. Robust correlation of encrypted attack traffic through stepping stones by manipulation of inter-packet delays. In *Proceedings of the 2003 ACM Conference on Computer and Communications Security (CCS)*, November 2003.
17. Wang X, Chen S, Jajodia S. Tracking anonymous peer-to-peer voip calls on the internet. In *Proceedings of the 12th ACM Conference on Computer Communications Security (CCS)*, November 2005.
18. Peng P, Ning P, Reeves DS. On the secrecy of timing-based active watermarking trace-back techniques. In *Proceedings of the IEEE Security and Privacy Symposium (S&P)*, May 2006.
19. June Pyun Y, Hee Park Y, Wang X, Reeves DS, Ning P. Tracing traffic through intermediate hosts that repacketize flows. In *Proceedings of IEEE INFOCOM*, May 2007.
20. Wong TF. Spread spectrum and code division multiple access. http://wireless.ece.ufl.edu/twong/notes1.html, August 2000.
21. ir. Meel J. Spread spectrum (SS)—introduction, http://www.sss-mag.com/pdf/Ss_jme_denayer_intro_print.pdf, 1999.
22. Oppenheim AV, Willsky AS, Nawab SH. Signals and systems, Prentice-Hall, Upper Saddle River, NJ 07458, USA, second edition, 1997.
23. Fu X, Yu W, Jiang S, Graham S, Guan Y. Tcp performance in flow-based mix networks: modeling and analysis. *IEEE Transactions on Parallel and Distributed Systems (TPDS)* 2009; **20**(5).