Blind Detection of Spread Spectrum Flow Watermarks

Weijia Jia, Fung Po Tso, Zhen Ling, Xinwen Fu, Dong Xuan and Wei Yu

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 due to 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 multi-flow-based approach in DSSS watermark detection. We also present theory for reconstructing the DSSS code once the DSSS code length is known and simulations validate the feasibility.

Index Terms

Anonymity, Detection, DSSS, Mean-square autocorrelation

Weijia Jia and Fung Po Tso are with City University of Hong Kong, Tat Chee Avenue, Kowloon, Hong Kong SAR (Email: {wei.jia, posco.tso}@cityu.edu.hk). Zhen Ling is with School of Computer Science and Engineering, Southeast University, Liwenzheng Building (North) #241, Nanjing 210096, P.R. China (Email: zhen_ling@seu.edu.cn). Xinwen Fu is with Department of Computer Science, University of Massachusetts Lowell, Lowell, MA 01854 (Email: xinwenfu@cs.uml.edu). Dong Xuan is with Department of Computer Science and Engineering, The Ohio-State University, Columbus, OH 43210 (Email: xuan@cse.ohio-state.edu). Wei Yu is with Department of Computer and Information Sciences, Towson University, Towson, MD 21252 (Email: wyu@towson.edu).

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

In recent years, significant progress has been made in the journey of fighting with attacks on anonymous communication networks. However, the journey is far from over [1], [2], [3], [4], [5], [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”) based on the same PN code. Spread spectrum techniques are resistant to interference and interception. This technique could be maliciously used by the attacker, in order to trace users of an anonymous communication network. In such attack, an 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 based on 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 since 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 kind of attack. In [8], Kiyavash et al introduced a multi-flow approach to detect DHSS watermarks. This approach requires a Markov-Modulated Poisson Process model.

In this paper, we introduce a statistical approach to detect DHSS watermarks and defeat 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 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 to detect DHSS watermarks. We also study the approach to accurately recover the DHSS code embedded in marked traffic. We evaluate the effectiveness of 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 of 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) while their approach does not. Our approach is based on a simple statistic, mean-square autocorrelation while their approach requires a Markov-Modulated Poisson Process model.
Once the malicious traceable has been detected, the victim sender or receiver may stop communication to thwart further detection. We also discuss the approach that enables the reconstruction of 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 intrusion reaction forensics is not within the scope of this paper.

The rest of the paper is organized as follows. In Section III, we review the DHSS-based traceable technique introduced in [7]. In Section IV, we introduce the mean-square autocorrelation and our approach 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 to reconstruct the DHSS code. In Section V, we use ns-2 simulations and network experiments over Tor to validate our findings, respectively. The disadvantage of the multi-flow detection approach in [8] is also presented in Section V-E. We review related work in Section II and conclude this paper in Section VI.

II. 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. To determine whether Alice is communicating with Bob, through a mix network, 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 packets, their approach can correlate encrypted network connections. Nevertheless, these timing-based schemes are not effective at tracing

---

1We use DHSS code and PN code interchangeably in this paper.
communication through a mix network with batching strategies that manipulate inter-packet delivery timing, as indicated in [7]. Peng et al. in [18] analyzed the secrecy of timing-based watermarking traceable proposed in [16], based on the distribution of traffic timing. Nevertheless, our focus on this paper is to detect the malicious DHSS-based flow marking technique proposed in [7].

Kiyavash, Houmansadr and Borisov [8] proposed a multi-flow approach 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 to recover the watermarking parameters and remove watermarks in the case of interval based watermarks. They applied a probabilistic model and the Markov-modulated Poisson process (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.

III. BACKGROUND

In this section, we first review the basic framework of the DHSS watermarking technique and then discuss the secrecy of this technique on how to escape detection. An introduction to the basic DHSS principle can be found in Appendix A.

A. Framework of DHSS Watermarking

The framework for the DHSS watermarking technique in [7] is illustrated in Figure 1. The basic idea is illustrated below. The malicious interferer first spreads each bit of a signal by 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 for 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, since longer signals decrease the false positive rate for traceable [7]. The transmitted baseband signal $X$ can be written as,

$$X = xC_t,$$  
(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,

$$T_x = Ax C_t + D. \tag{2}$$

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. Denoting noise as a random variable $\xi$, we can formulate the received signal $R_x$ as,

$$R_x = Ax C_t + D + \xi. \tag{3}$$

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 $\bar{R}_x$. All items in Equation (3) are $1 \times l$ vectors, where $l$ is the PN code length, i.e., 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,

$$R_x' \approx Ax C_t + \xi. \tag{4}$$
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 $R_b$,

$$R_b = R'_x \cdot C_r = A d_t C_t \cdot C_r + \xi \cdot C_r,$$

(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, to recognize the original signal $x$, the DHSS watermarking requires that the locally generated PN code at the sniffer is synchronized with the one at the interferer. 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.

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. 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.

### B. Secrecy of DHSS Watermarking

Secrecy of DHSS watermarks refers to the difficulty of detecting the traceable and 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 based on the secrecy of the code (including the chip duration). Since we don’t 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 Formula (2) can be very small in comparison with noise so that the DHSS mark $X$ is covered by noise $\xi$ 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 don’t 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 random in order to maintain the secrecy. It is not feasible to recognize PN code modulated traffic in both time and frequency domains.

### IV. 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 mean-square autocorrelation (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 DHSS code and be used for misleading the malicious traceable.

To simplify our analysis, we assume that the chip duration is 1 unit (e.g., 1 second) 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 non-synchronized case, where a window doesn’t necessarily begin (or end) on a bit boundary. Finally we describe the workflow of detecting DHSS watermarks, introduce an automatic decision rule and discuss related issues.

A. Mean-Square Autocorrelation in a Synchronized Window

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

![Fig. 2: Self-similarity of PN code Modulated Traffic (first row - a traffic segment; second row - the time shifted version of the segment)](image)

![Fig. 3: a Non-Synchronized Window (first row - a traffic segment; second row - the time shifted version of the segment)](image)
Therefore, the modulated signal $\tilde{X}$ can be written as follows:

\[
\tilde{X} = (x_0 \vec{C}, x_1 \vec{C}, \cdots, x_{w-1} \vec{C}),
\]

\[
= (x_0 c_0, \cdots, x_0 c_l-1, \cdots, x_{w-1} c_0, \cdots, x_{w-1} c_l-1)
\]

(6)

where $\tilde{X}$ 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 (iid). 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 standard deviation $\sigma = A$. We can use Formula (8) to estimate the autocorrelation of a time series represented by $\tilde{X}$,

\[
r(\tau) = \frac{1}{(wl - \tau)} \sum_{i=1}^{wl-\tau} y_i y_{i+\tau},
\]

(8)

In (8), $\tau$ is the lag and $y_i = x_{[i/l]} c_{i \% l}$ is the $i^{th}$ item of $\tilde{X}$, where $[i/l]$ 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 (7) and (8),

\[
r(kl) = \frac{1}{wl - kl} \sum_{i=0}^{wl-kl-1} y_i y_{i+kl}
\]

\[
= \frac{1}{wl - kl} (x_0 x_0 + k \vec{C} \cdot \vec{C} + \cdots + x_{w-1-k} x_{w-1-k} \vec{C} \cdot \vec{C}),
\]

(9)

where $\cdot$ refers to dot product and $\vec{C} \cdot \vec{C} = l$. Therefore,

\[
r(kl) = \frac{1}{w - k} (x_0 x_0 + \cdots + x_{w-1-k} x_{w-1}).
\]

(10)

$r^2(\tau)$ is the square autocorrelation of spread signal $\tilde{X}$ and a time-shifted $\tilde{X}$ with lag $\tau$. 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. Since we don’t have access to the code, we can’t 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. Since 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 a precise multiple of the PN code length and correlated, the autocorrelation yields a non-zero 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 $\tau$ $(0 \leq \tau < wl)$,

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

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 Formula (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 (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.

- 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 code, even cryptographically secure pseudorandom number generators.

- $E(r^2(\tau))$ of the PN code modulated traffic shows peaks of value $\frac{A^4}{w-k}$ when $\tau = kl, 1 \leq k < w$. $k$ is the time-shift factor and corresponds to the number of complete bits shifted before calculating the MSAC. Larger the $k$, the higher the peak value. The highest peaks occur when $\tau = 0$ and $\tau = (w-1)l$.

- We can infer the code length $l$ based on 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 details in Section IV-C. We give a simple example to illustrate how to calculate MSAC in Appendix C.

**B. Mean-Square Autocorrelation (MSAC) in a Non-Synchronized 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 $l = 5$ and we have fourteen 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 3 chips of modulated bit $x_0$, 2 complete modulated bits $x_1$ and $x_2$, and 1 chip of modulated bit $x_3$. When lag $\tau \neq kl$, the non-synchronized PN codes in the original traffic segment and its time-shifted version produces a minimal $r(\tau)$, and thus a minimal
When $r^2(\tau)$ (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 non-synchronized window. Its proof can be found in Appendix D.

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

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

where $k$ is an integer.

We have a few observations from Corollary 2:

- The MSAC of DHSS watermarks in a non-synchronized window still demonstrates periodicity. The code length $l$ is equal to the interval between two consecutive peaks of the MSAC.
- 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.

### C. Framework of Detecting DHSS Watermarks

In Section IV-A, we demonstrated that the MSAC of DHSS watermarks shows periodicity and may be used for 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.

![ Workflow of Detecting DHSS Watermarks](image)

**Fig. 4: Workflow of Detecting DHSS Watermarks**

Figure 4 shows the four stages of 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. Since the detection scheme is based on the autocorrelation of multiple-bits DHSS watermarks, the acquired traffic segments must be longer than one signal bit. Based on 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 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 for determining $t_a$. In our experiments, 0.1s is a good selection for $t_a$. In the practical blind detection of DHSS watermarks, the unit of $\tau$ in Formula (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 since our analysis in Section IV-A is for data in the bipolar format. We can use the Fast Fourier Transform (FFT) to achieve this: (a) calculate the FFT of the time series, (b) change the frequency component at zero frequency to zero in order to remove the direct component, and (c) 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 (14),

$$E(r^2(\tau)) \approx \frac{1}{M} \sum_{i=1}^{M} r_i^2(\tau),$$

(14)

where $M$ is the number of segments and $r_i^2(\tau)$ is the MSAC at lag $\tau$ for the $i^{th}$ 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.

D. 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 for 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 $\tau$ 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 $E(r^2(k\tau)) > E(r^2(\tau))$, where $\tau \neq k\tau$, peaks will appear. Thus, $E(r^2(\tau))(\tau \neq k\tau)$ can be viewed as “noise” and $E(r^2(k\tau))$ 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,

$$y_i = x_i + \xi_i,$$

(15)

where $x_i$ is the random variable of the modulated signal and $\xi_i$ 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,

$$ E(r_x^2(\tau)) = \begin{cases} \delta^4 \frac{1}{1/wl + 1}, & \text{if } \tau = 0, \\ \frac{\delta^4}{wl - \tau}, & \text{if } \tau \neq 0. \end{cases} $$

(16)

where $\delta^2$ is the noise variance.

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

**Theorem 3:** The signal-to-noise ratio for the MSAC can be estimated as follows,

$$ \frac{E(r_x^2(\tau))}{E(r_x^2(\tau))} = \begin{cases} \frac{4^*}{\sigma^2} / \frac{1}{wl + 1}, & \text{if } \tau = 0, \\ l \frac{4^*}{\sigma^2}, & \text{if } \tau = kl, 1 \leq k < w, \\ 0, & \text{if } \tau \neq kl, 1 \leq k < w. \end{cases} $$

(17)

**Proof:** Based on $E(r_x^2(\tau))$ in (12) of Theorem 1 and $E(r_x^2(\tau))$ in (16) of Theorem 2, the signal-to-noise ratio 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 $l \frac{4^*}{\sigma^2}$ when $\tau = kl$, where $0 < k < w$. Moreover, the longer the code length $l$, the higher the peak. Therefore, a longer code makes recognition of DHSS watermarks easier.

We introduce an automatic rule in Algorithm 1 for detecting DHSS watermarks via periodicity of peaks in $E(r_x^2(\tau))$. 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 signal bit. Finally, given a false positive rate, we use hypothesis testing to make 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 $\eta$ in Algorithm 1 and derive the corresponding detection rate and false positive rate for each case. It will be shown that this algorithm is effective since the detection rate can be high while the false positive rate is kept small.

**E. Parameters Affecting the Blind Detection of Spread Spectrum Flow Watermarks**

We now discuss the impact of PN code length and number of signal bits on the blind detection of spread spectrum flow watermarks. Based on results shown in Theorem 3, we know that the SNR increases linearly with 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) $\eta$, 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 $\mu_\xi$, the mean of noise MSAC; $\delta_\xi$, standard deviation of $r_\xi^2(\tau)$, where $\tau \neq kl t_a$ and $\tau < (w-1)lt_a$
5: Calculate the estimated signal MSAC $r_{x;i}^2 = \frac{1}{w-1} \sum_{k=1}^{w-1} r_\xi^2(klt_a)$ for the $i^{th}$ traffic segment
6: if $\frac{1}{M} \sum_{i=1}^{M} r_{x;i}^2 > \mu_\xi + \eta \delta_\xi$ then
7: return result $\leftarrow$ true
8: end if
9: end while
10: return result $\leftarrow$ false

The longer the code length, the higher the SNR and the easier 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 relative longer code length.

Recall that since our detection relies on the autocorrelation of DHSS watermarks, the DHSS watermarks must contain multiple bits ($\geq 2$). 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 traceable will be too significant at 50% [7]. Given a target false positive rate of 1%, the number of bits must be larger than 7. This will favor the detection of DHSS watermarks again. We will show in Sections V and V-F that 7 bits or fewer is enough for detecting DHSS watermarks.

F. Reconstruction of DHSS Code

As we can see, 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 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 IV-C.

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. Since 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

\[
y = x_k \tilde{C}_e + x_{k+1} \tilde{C}_s + \tilde{\xi},
\]

where vectors \( \tilde{C}_e \) and \( \tilde{C}_s \) have a length of \( l \), \( \tilde{\xi} \) is the noise random variable and

\[
\tilde{C}_e = \{ \text{the last } l_0 \text{ chips of } x_k, \text{ and } l - l_0 \text{ 0's} \},
\]

\[
\tilde{C}_s = \{ l_0 \text{ 0's, and the first } l - l_0 \text{ chips of } x_{k+1} \}.'
\]

where symbol ' refers to the matrix transpose. Therefore, \( \tilde{C}_e \) and \( \tilde{C}_s \) are column vectors.

Then we can derive the correlation matrix \( R_y \) of \( y \).

\[
R_y = E \left( (x_k \tilde{C}_e + x_{k+1} \tilde{C}_s + \tilde{\xi}) \right) \left( (x_k \tilde{C}_e + x_{k+1} \tilde{C}_s + \tilde{\xi})' \right),
\]

\[
= E(x_k^2 \tilde{C}_e * \tilde{C}_e' + x_k x_{k+1} \tilde{C}_e * \tilde{C}_s' + x_k \tilde{C}_e * \tilde{\xi}')
+ x_{k+1} x_k \tilde{C}_s * \tilde{C}_e' + x_k^2 \tilde{C}_s * \tilde{C}_s' + x_{k+1} \tilde{C}_e * \tilde{\xi}'
+ x_k \tilde{\xi} * \tilde{C}_e' + x_{k+1} \tilde{\xi} * \tilde{C}_s' + \tilde{\xi} * \tilde{\xi}'
\]

\[
= E(x_k^2 \tilde{C}_e * \tilde{C}_e' + E(x_{k+1}^2) \tilde{C}_s * \tilde{C}_s' + E(\tilde{\xi} * \tilde{\xi}')
+ E(\tilde{\xi} * \tilde{\xi}')
\]

\[
= A^2 \tilde{C}_e * \tilde{C}_e' + A^2 \tilde{C}_s * \tilde{C}_s' + \sigma^2 I.
\]

where \( I \) is the identity matrix.

In the following, we try to derive the eigenvalues and eigenvectors of \( R_y \). From the construction of \( \tilde{C}_e \) and \( \tilde{C}_s \),
we can derive
\[
\mathbf{R}_y \vec{C}_e = (A^2 \vec{C}_e * \vec{C}_e' + A^2 \vec{C}_s * \vec{C}_s' + \sigma^2 \mathbf{I}) \vec{C}_e, 
\]
\[
= A^2 \vec{C}_e * \vec{C}_e' + A^2 \vec{C}_s * \vec{C}_s' + \sigma^2 \mathbf{I} \vec{C}_e, 
\]
\[
= A^2 \vec{C}_e l_0 + 0 + \sigma^2 \vec{C}_e, \quad (25)
\]
\[
= (A^2 l_0 + \sigma^2) \vec{C}_e, 
\]
\[
= \sigma^2 (1 + A^2 \frac{\sigma}{\sigma^2} l_0) \vec{C}_e. \quad (29)
\]

Denote \( \rho = \frac{A^2}{\sigma^2} \) as SNR (signal to noise ratio), and we have
\[
\mathbf{R}_y \vec{C}_e = \lambda \vec{C}_e = \sigma^2 (1 + \rho l_0) \vec{C}_e. \quad (30)
\]
Hence, one eigenvector of \( \mathbf{R}_y \) is \( \vec{C}_e \), and the corresponding eigenvalue is \( \lambda_e \) as follows,
\[
\lambda_e = \sigma^2 (1 + \rho l_0). \quad (31)
\]
Using the similar approach, we can also derive a second eigenvector \( \vec{C}_s \), and its eigenvalue is \( \lambda_s \) as follows,
\[
\lambda_s = \sigma^2 (1 + \rho (l - l_0)). \quad (32)
\]
In order to derive all the eigenvalues \( \lambda \) of \( \mathbf{R}_y \) in (24), we need to solve the following equation
\[
det(\mathbf{R}_y - \lambda \mathbf{I}) = 0, \quad (33)
\]
where \( det(.) \) refers to the calculation of matrix determinant.

**Theorem 4:** The determinant of \( \mathbf{R}_y - \lambda \mathbf{I} \) is given in (34),
\[
det(\mathbf{R}_y - \lambda \mathbf{I}) = (-1)^l (\lambda - \lambda_e)(\lambda - \lambda_s)(\lambda - \sigma^2)^{l-2}. \quad (34)
\]
Therefore, \( \mathbf{R}_y \) has three eigenvalues: \( \lambda_e, \lambda_s \) and \( \sigma^2 \), where \( \lambda_e, \lambda_s \geq \sigma^2 \). The two eigenvectors \( \lambda_e \) and \( \lambda_s \) corresponding to the two biggest eigenvalues \( \vec{R}_e \) and \( \vec{R}_s \) compose the DHSS code as shown in (19) and (20).

The proof of Theorem 4 is given in Appendix F. From Theorem 4, we can derive the DHSS code from eigenvectors \( \vec{C}_e \) in (19) and \( \vec{C}_s \) in (20) corresponding to the two principal eigenvalues \( \lambda_e \) and \( \lambda_s \). Since \( \mathbf{R}_y \) is a symmetric real matrix, we can diagonalize it into the following matrix \( \mathbf{A} \),
\[
\mathbf{A} = \begin{pmatrix}
\sigma^2 (1 + \rho l_0) & 0 & 0 & \cdots & 0 \\
0 & \sigma^2 (1 + \rho (l - l_0)) & 0 & \cdots & 0 \\
0 & 0 & \sigma^2 & \cdots & 0 \\
\cdots & \cdots & \cdots & \cdots & \cdots \\
0 & 0 & 0 & 0 & \sigma^2
\end{pmatrix} \quad (35)
\]
Using the results obtained above, we can apply the following steps to derive the original DHSS code practically:

(i) deriving the correlation matrix $R_y$; (ii) solving $R_y$ for its eigenvalues and eigenvectors; (iii) deriving the DHSS code: the two eigenvectors $\vec{R}_e$ and $\vec{R}_s$ corresponding to the two biggest eigenvalues compose the DHSS code as stated in Theorem 4. $\vec{R}_s$'s first few elements will be very small and treated as zeros. Its other elements are treated as 0s (if the element $< 0$) or 1s (if the element $> 0$) and compose the first part of the PN code. Similarly, the non-trivial elements of $\vec{R}_e$ compose the remaining part of the PN code. In Section V-D, we will present simulation results and validate the feasibility of this approach to reconstruct DHSS code.

V. 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 multi-flow detection approach in [8] is presented in Section V-E. Results of empirical tests over Tor will be presented in Section V-F. 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 summarize the trend of detection rate and false positive rate in terms of number of segments, window size, PN code length and signal to noise ratio (SNR).

A. 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 since they are not practical [23]) as used in [7]. The target 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 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 while 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.

B. Evaluation Metric

We use detection rate $P_D$ and false positive rate $P_F$ as our evaluation metrics for detecting DHSS watermarks. Recall 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 mean-square autocorrelation (MSAC) distribution, $f_1(x)$ the signal MSAC distribution, and $\gamma$ is determined by $\eta$ in Algorithm 1. 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 $\gamma$ increases, while both $P_D$ and $P_F$ increase to one as $\gamma$ 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 DHSS watermarks, we want a high detection rate and a low false positive rate.

![Fig. 6: Calculation of $P_D$ and $P_F$](image)

C. Detecting DHSS Marks

Now we show the MSAC based approach is effective for detecting DHSS watermarks. Figure 7 uses Formula (14) to estimate the MSAC. In this case, the PN code length is 7. Only 2 traffic segments are used and each segment contains about 3 bits. So totally at most 6 bits are used in this non-synchronized case. The chip duration $t_c$ is 0.5s and sampling interval is 0.1s. The interference traffic CBR traffic rate is 1.2Mbps. We have a couple of observations from Figure 7. First, peaks indeed appear at multiples of the bit period. The bit length is $7 \times 0.5 = 3.5s$. Second, false positives may occur since there are some peaks at un-expected places.

Figure 8 shows the detection rate and false positive rate in terms of the number of segments. Other parameters are the same as above. Figure 9 shows the ROC curve when the number of segments is 4. We have the following observations from Figure 8 and Figure 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 IV. Since 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% while 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 2. 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 IV-D. As the window size increases, there will be more possible peaks at expected positions. Since our decision rule considers the effect of all those peaks, the 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 number of segments is 2 and window size is 3. Based on Theorem 3, we know when 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 7, the number of segments is 4 and window size is 3. When the interference intensity increases, SNR increases generally. A larger SNR will reduce false positive rate and increase detection rate as shown in Figure 12.

Figure 13 summarize the trend of detection rate and false positive rate in terms of number of segments, window size, PN code length and signal to noise ratio (SNR). Symbol ↗ refers to the increasing function. For example, detection rate is an increasing function of number of segments. Symbol ↘ refers to the decreasing function. For example, false positive rate is a decreasing function of number of segments.

<table>
<thead>
<tr>
<th></th>
<th>Detection rate</th>
<th>False positive rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of segments</td>
<td>↗</td>
<td>↘</td>
</tr>
<tr>
<td>Window size</td>
<td>↗</td>
<td>↘</td>
</tr>
<tr>
<td>PN code length</td>
<td>↗</td>
<td>↘</td>
</tr>
<tr>
<td>Signal to noise (SNR) ratio</td>
<td>↗</td>
<td>↘</td>
</tr>
</tbody>
</table>

Fig. 13: Summary of Detection Rate and False Positive Rate Trend

Fig. 14: Reconstruction of PN Code of DHSS-based Traceable

D. Reconstruction of DHSS Code

In Section IV-F, we discussed the reconstruction of the DHSS code in case of Gaussian noise and random signal. We conduct simulations within Matlab and validate the feasibility of the theory. Figure 14 illustrates the recovery ratio versus the signal to noise ratio (SNR). The DHSS code is 7 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 m-sequence code, they may utilize the properties of such 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 (24). A more accurate correlation matrix produces more accurate eigenvectors that compose the DHSS code by utilizing Theorem 4.

E. Deficiency of Multi-flow Detection in [8]

In [8], a multi-flow 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 Markov Modulated Poisson Process (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 $P_{tc}$. When the number of flows increases and $P_{tc}$ is smaller than a threshold, they can decide whether the traffic is watermarked.

Based on their Formula (6) in [8], we draw $P_{tc}$ in terms of the number of flows and interval of low traffic rate in Figure 15. The MMPP model used for Figure 15 is trained using ftp downloading sessions over Tor. We can see that the multi-flow detection approach cannot detect a DHSS watermarked flow when the chip duration $t_c$ varies from 0.1s to 1.0s and there is only one watermarked flow, given a threshold of 1%. When $t_c = 0.2s$, 8 flows are required to detect watermarked flows. When $t_c = 0.4s$, 4 flows are required. Requiring multiple simultaneous watermarked flow 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.

F. 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. 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 with appropriate proxy configuration in order to use Tor.

In order to carry out 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 based on 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, $t_c = 3$ seconds and the code length, $l = 7$. So the bit duration is 21 seconds as used in [7]. From Figure 17, we can see that the MSAC indeed demonstrates periodicity at positions of multiple 21 seconds, and we can effectively detect DSSS watermarks.

VI. Conclusion

In this paper, we proposed an approach for blindly detecting malicious DSSS traceback, which may be applied to trace anonymous traffic flows and seriously degrade the anonymity that an anonymous communication network
provides. By calculating the mean-square autocorrelation (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 theory to recover the DSSS code and our simulations verified the recovery theory.

ACKNOWLEDGMENT

The 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, CNS-1117175, CAREER Award CCF 0546668, by the Army Research Office (ARO) under grant No. AMSRD-ACC-R50521-CI, and by the Army Research Laboratory under grand No. W911NF-11-1-0193. 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.

REFERENCES

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 receiver. Each chip in the PN code lasts for \( t_c \) seconds (denoted as \textit{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 \textit{PN code length}. These concepts are illustrated in Figure 19.

![Figure 19: Direct Sequence Spread Spectrum (DSSS)](image)

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.

![Figure 19: Spreading and Despreading in DSSS](image)

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 \( X \). To recover the original signal from \( X \), \( X \) is multiplied with the same PN code at the receiver. We have the recovered signal

\[
x = \sum_{l} (X \cdot C) = x \sum_{l} (C \cdot C),
\]

where the operator of \( \cdot \) 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, but 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 (7) and (8), we have

$$ r(0) = \frac{1}{wl} \sum_{i=0}^{w-1-l-1} x_i^2 c_i^2. \quad (37) $$

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

$$ r(0) = A^2, \quad (38) $$

and

$$ E(r^2(0)) = A^4. \quad (39) $$

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

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

$$ = \frac{1}{wl - kl} (x_0 x_{k} + \cdots + x_{w-1-k} x_{w-1} \bar{C} \cdot \bar{C}) $$

$$ + \cdots + x_{w-1-k} x_{w-1} \bar{C} \cdot \bar{C}), \quad (41) $$

where \( \cdot \) refers to dot product and $\bar{C} \cdot \bar{C} = l$. Therefore,

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

$$ = \frac{1}{w - k} \sum_{i=0}^{w-1-k} x_i x_{i+k}. \quad (43) $$

The mean square autocorrelation $E(r^2(kl))$ can be calculated as follows:

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

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

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

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

$$ = \frac{1}{(w - k)^2} E( \sum_{i=0}^{w-1-k} \sum_{j=0}^{w-1-k} x_i x_{i+k} x_j x_{j+k}) \quad (47) $$
\[
E(\sum_{i=j}^{\infty} x_i x_{j+k}) + \sum_{i \neq j} x_i x_{i+k} x_{j+k},
\]
(48)
\[
= \frac{1}{(w-k)^2} \sum_{i=0}^{w-1-k} E(x_i^2 x_{i+k}) + \sum_{i \neq j} E(x_i x_{i+k} x_{j+k}).
\]
(49)

Since one of \(x_i, x_{i+k}, x_j\) and \(x_{j+k}\) (e.g., \(x_i\)) will be independent from the other three random variables and
\(E(x_i) = 0, \sum_{i \neq j} E(x_i x_{i+k} x_{j+k}) = 0\). Therefore,
\[
E(r^2(kl)) = \frac{A^4}{(w-k)^2} (w-k+0),
\]
(50)
\[
= \frac{A^4}{w-k}.
\]
(51)

**Case 3** \(\tau \neq kl, 0 \leq k < w\): In this case, the autocorrelation can be calculated as follows
\[
r(\tau) = \frac{1}{(wl-\tau)} \sum_{i=0}^{wl-1-\tau} x_{[i/|l|]} c_{i|l|} x_{[(i+\tau)/|l|]} c_{(i+\tau)|l|},
\]
(52)
\[
= \frac{1}{(wl-\tau)} \sum_{i=0}^{wl-1-\tau} x_{[i/|l|]} x_{[(i+\tau)/|l|]} c_{i|l|} c_{(i+\tau)|l|}.
\]
(53)

(53) 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 \(\tau\) is non-zero, 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
\[
E(x_i x_j c_i c_{i+\tau}) = E(x_i x_j) E(c_i c_{i+\tau}) = 0
\]
(54)
\[
r(\tau) = 0, \tau \neq kl, 0 \leq k < w.
\]
(55)

Therefore,
\[
E(r^2(\tau)) = 0, \tau \neq kl, 0 \leq k < w.
\]
(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 7 bits, all 1’s, \(\{1, 1, 1, 1, 1, 1, 1\}\) and the PN code has 7 chips \(\{1, -1, -1, 1, 1, 1, -1\}\). Therefore, the spread signal \(X\) is
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. Formula (8), repeated here, estimates the autocorrelation of one sample.

\[
\begin{align*}
\text{(57)} \\
1, -1, -1, 1, 1, -1, \\
1, -1, -1, 1, 1, -1, \\
1, -1, -1, 1, 1, -1, \\
X &= 1, -1, -1, 1, 1, -1, \\
1, -1, -1, 1, 1, -1, \\
1, -1, -1, 1, 1, -1, \\
1, -1, -1, 1, 1, -1
\end{align*}
\]

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

\[
\begin{align*}
\text{(58)} \\
\rho(\tau) &= \frac{1}{(wl - \tau)} \sum_{i=1}^{wl-\tau} y_i y_{i+\tau}
\end{align*}
\]

Then, we can use two samples to calculate an average for \( r^2(\tau) \), i.e., an estimation of \( E(r^2(\tau)) \). Rows 1 and 2 in (57) constitute our first sample and rows 3 and 4 constitute our second sample. By simple calculation, we can get the estimated \( E(r^2(\tau)) \) as illustrated in Figure 20. We can see that the period of the peaks is indeed 7 (the PN code length) as Theorem 1 indicates.

Fig. 20: A Simple Example of Calculating \( E(r^2(\tau)) \)

APPENDIX D

In this appendix, we prove Corollary 2.

\textit{Proof:}

Let’s derive an approximate estimation of \( r^2(\tau) \) for 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

\[
0 < s + e < 2l,
\]

\[
w l + \Delta = s + w'l + e. \tag{59}
\]

Rearranging (60), we have

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

Since \( 0 < \Delta < l \),

\[
-2l < -(s + e) < \Delta - (s + e) < l - (s + e) < l, \tag{62}
\]

we know

\[
w l - 2l < w'l < wl + l. \tag{63}
\]

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. For the case of traffic segments containing \( w \) or \( w - 1 \) bits, we can use Theorem 1 to estimate their MSAC, \( r^2_w(\tau) \) and \( r^2_{w-1}(\tau) \), respectively. Then utilizing the Total Probability Theorem, we know

\[
r^2(\tau) = pr^2_w(\tau) + qr^2_{w-1}(\tau) \tag{64}
\]

where \( r^2_w(\tau) \) and \( r^2_{w-1}(\tau) \) can be calculated in (12).

Therefore, assuming \( k \) is an integer, we have:

- When \( \tau = 0 \), \( r^2(0) = pr^2_w(0) + qr^2_{w-1}(0) = A^4 \).
- When \( \tau = kl \) and \( 0 < k < w - 1 \), \( r^2(\tau) = pr^2_w(\tau) + qr^2_{w-1}(\tau) = p A^4 + q A^4 \).
- When \( \tau = (w - 1)l \), that is \( k = w - 1 \), \( r^2_{w-1}(\tau) = 0 \), so that \( r^2(\tau) = pr^2_w(\tau) + qr^2_{w-1}(\tau) = p A^4 \).
- When \( \tau \neq kl \) and \( 0 \leq k < w \), \( r^2(\tau) = pr^2_w(\tau) + qr^2_{w-1}(\tau) = p \times 0 + q \times 0 = 0 \).

Summarizing the results above, we derive (13) 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 \( \xi_i \) the random variable of noise.
We can calculate the mixed signal $y$'s autocorrelation as follows,

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

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

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

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

Apply Theorem 1 to (68), we have

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

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

Noise time series $\vec{\xi}$ can be represented as follows,

$$\vec{\xi} = (\xi_1, \cdots, \xi_{wl}).$$

Then the correlation of $\vec{\xi}$ can be estimated as follows,

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

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

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

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

If $\tau = 0$, we have

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

$$= \frac{1}{(wl)^2} (wl^2\delta^4 + (wl)^2\delta^4),$$

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

If $\tau \neq 0$, we have

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

$$= \frac{\delta^4}{wl-\tau}.$$
In this appendix, we prove Theorem 4. Let $C_e$ and $C_s$ have a format as in (79) and (80),

$$
\tilde{C}_e = \begin{bmatrix} 1 & -1 & 1 & 0 & 0 \end{bmatrix}',
$$

(79)

$$
\tilde{C}_s = \begin{bmatrix} 0 & 0 & 0 & -1 & 1 \end{bmatrix}'.
$$

(80)

$R_y$ can be written as follows,

$$
R_y = \begin{pmatrix} R_U & 0 \\ 0 & R_L \end{pmatrix}
$$

(81)

In particular, in agreement to $\tilde{C}_e$ and $\tilde{C}_s$ in (79) and (80), we have

$$
R_U = \begin{pmatrix} A^2 + \sigma^2 & -A^2 & A^2 \\ -A^2 & A^2 + \sigma^2 & -A^2 \\ A^2 & -A^2 & A^2 + \sigma^2 \end{pmatrix}
$$

(82)

and

$$
R_L = \begin{pmatrix} A^2 + \sigma^2 & -A^2 \\ -A^2 & A^2 + \sigma^2 \end{pmatrix}
$$

(83)

Denote the non-zero items of $\tilde{C}_e$ and $\tilde{C}_s$ as $\tilde{C}_{eu}$ and $\tilde{C}_{sl}$ and we know

$$
R_U = A^2 \tilde{C}_{eu} \tilde{C}_{eu}' + \sigma^2 I, 
$$

(84)

$$
R_L = A^2 \tilde{C}_{sl} \tilde{C}_{sl}' + \sigma^2 I.
$$

(85)

Therefore, we have

$$
det(R_y - \lambda I) = det(R_U - \lambda I) \times det(R_L - \lambda I).
$$

(86)

Now let us calculate $det(R_U - \lambda I)$. In order to do so, we first linearly transform $R_U - \lambda I$ and we have

$$
R_U - \lambda I = A^2 \tilde{C}_{eu} \tilde{C}_{eu}' + \sigma^2 I - \lambda I, 
$$

(87)

$$
= A\tilde{C}_{eu} (A\tilde{C}_{eu}')' + (\sigma^2 - \lambda) \tilde{I}.
$$

(88)

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

$$
R_U - \lambda I = \begin{pmatrix} b_{11}B \\ b_{12}B \\ \vdots \\ b_{1l_0}B \end{pmatrix} + (\sigma^2 - \lambda) \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}
$$

(89)
We can linearly transform $R_y^U - \lambda I$ as follows,

$$
R_y^U - \lambda I \rightarrow \begin{pmatrix}
0 \\
0 \\
\vdots \\
0
\end{pmatrix} + (\sigma^2 - \lambda) 
\begin{pmatrix}
1 & 0 & \cdots & -\frac{b_{11}}{b_{110}} \\
0 & 1 & \cdots & -\frac{b_{12}}{b_{110}} \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 1
\end{pmatrix}
$$

(90)

$$
= \begin{pmatrix}
\sigma^2 - \lambda & 0 & \cdots & -\frac{b_{11}}{b_{110}}(\sigma^2 - \lambda) \\
0 & \sigma^2 - \lambda & \cdots & -\frac{b_{12}}{b_{110}}(\sigma^2 - \lambda) \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & \sigma^2 - \lambda + b_{11}^2
\end{pmatrix}
$$

(91)

We can further transform (91) as follows,

$$
R_y^U - \lambda I \rightarrow \begin{pmatrix}
\sigma^2 - \lambda & 0 & \cdots & -\frac{b_{11}}{b_{110}}(\sigma^2 - \lambda) \\
0 & \sigma^2 - \lambda & \cdots & -\frac{b_{12}}{b_{110}}(\sigma^2 - \lambda) \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & \sigma^2 - \lambda + b_{11}^2 + b_{12}^2
\end{pmatrix}
$$

(92)

$$
\rightarrow \begin{pmatrix}
\sigma^2 - \lambda & 0 & \cdots & -\frac{b_{11}}{b_{110}}(\sigma^2 - \lambda) \\
0 & \sigma^2 - \lambda & \cdots & -\frac{b_{12}}{b_{110}}(\sigma^2 - \lambda) \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & \sigma^2 - \lambda + b_{11}^2 + b_{12}^2
\end{pmatrix}
$$

(93)

$$
\rightarrow \begin{pmatrix}
\sigma^2 - \lambda & 0 & \cdots & -\frac{b_{11}}{b_{110}}(\sigma^2 - \lambda) \\
0 & \sigma^2 - \lambda & \cdots & -\frac{b_{12}}{b_{110}}(\sigma^2 - \lambda) \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & \sigma^2 - \lambda + \sum_{i=1}^{l_0} b_{1i}^2
\end{pmatrix}
$$

(94)

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

$$
R_y^U - \lambda I \rightarrow \begin{pmatrix}
\sigma^2 - \lambda & 0 & \cdots & -\frac{b_{11}}{b_{110}}(\sigma^2 - \lambda) \\
0 & \sigma^2 - \lambda & \cdots & -\frac{b_{12}}{b_{110}}(\sigma^2 - \lambda) \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & \sigma^2 - \lambda + \sum_{i=1}^{l_0} b_{1i}^2 + A^2
\end{pmatrix}
$$

(95)

Therefore,

$$
\det(R_y^U - \lambda I) = (-1)^{l_0}(\lambda - \sigma^2)^{l_0-1}(\lambda - (\sigma^2 + l_0 A^2)).
$$

(96)
Similarly, we can derive

\[
det(R^L_y - \lambda I) = (-1)^{l-l_0}(\lambda - \sigma^2)^{l-l_0-1} (\lambda - (\sigma^2 + (l-l_0)A^2)).
\]  
(97)

Finally, we have

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

\[
= (-1)^l(\lambda - \sigma^2)^{l-2} (\lambda - \sigma^2(1 + l_0A^2/\sigma^2)) (\lambda - \sigma^2(1 + (l-l_0)A^2/\sigma^2)),
\]  
(99)

\[
= (-1)^l(\lambda - \sigma^2)^{l-2} (\lambda - \lambda_e) (\lambda - \lambda_0).
\]  
(100)