Blind Detection of Spread Spectrum Flow Watermarks

Weijia Jia[†], Fung Po Tso[†], Zhen Ling[£], Xinwen Fu[‡], Dong Xuan[§] and Wei Yu[*]

[†] City University of Hong Kong, Tat Chee Avenue, Kowloon, Hong Kong SAR

[£] Southeast University, Nanjing 210096, P. R. China

[‡] University of Massachusetts, Lowell, MA 01854

[§] The Ohio-State University, Columbus, OH 43210

[*] Cisco Systems, Inc., TX 75082

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 a simple 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 and real-world experiments on Tor. Our results demonstrate a high detection rate with a low false positive rate. Our scheme is more flexible and accurate than an existing multi-flow-based approach in DSSS watermark detection.

I. INTRODUCTION

In recent years, significant progress has been made in the journey of fighting attacks against anonymous communication networks. However, the journey is far from over [1], [2], [3], [4], [5]. Significant research efforts are still needed to discover and counter these attacks.

A new type of attack against anonymous communication networks has been discovered recently. In [6], Yu *et al.* proposed a direct sequence spread spectrum (DSSS) based technique, which could be used to maliciously trace users of an anonymous communication networks. In the traceback attack, at the sender side, an attacker called *interferer* modulates (spreads) a multi-bit signal with a single pseudo-noise (PN) code, interferes with the target traffic and embeds the modulated signal into the victim sender's outbound traffic flow rate pattern. This process is analogous to watermarking the target traffic. At the receiver side, another attacker called *sniffer* can analyze the victim receiver's inbound traffic flow rate time series to recover the modulated signal with noise, and demodulate (despread) the modulated signal to derive the original multi-bit signal based on the same secret PN code. The secret PN code is known to only the interferer and sniffer. In this way, the attackers can confirm the communication

relationship between the sender and the receiver.

The above attack is difficult to detect since DSSS modulated traffic shows a white noise-like pattern in both frequency and time domains. Some efforts have been made to detect such attack. In [7], Kiyavash *et al.* introduced a multi-flow approach detecting DSSS watermarks. The approach requires a Markov-Modulated Poisson Process model, which is complicated and whose parameters are difficult to derive for the Internet traffic. The detection requires multiple watermarked flows and this constraint is not easy to meet in reality either.

In this paper, we introduce a simple statistical approach to detect the existence of DSSS watermarks and defeat malicious traceback. Recall in the DSSS 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 traceback activity. We conduct a thorough theoretic analysis of this approach detecting DSSS watermarks. We evaluate this detection scheme's effectiveness 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 DSSS watermarks is simple, efficient and effective compared with the approach in [7]. It is based on a simple statistic, mean-square autocorrelation, and can deal with a single flow (or a few flows).

Once the malicious traceback can be detected, the victim sender or receiver may stop communication to thwart traceback. 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 traceback. The part of intrusion reaction and forensics is not within the scope of this paper.

The rest of the paper is organized as follows. In Section II, we review the DSSS-based traceback technique introduced in [6]. In Section III, we introduce the mean-square auto-correlation and our approach detecting a malicious DSSS

based traceback by calculating MSAC of the DSSS modulated traffic. In Section IV and 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 [7] is presented in Section IV-D. We review related work in Section VI and conclude this paper and discuss the future work in Section VII.

II. BACKGROUND

In this section, we first review the basic framework of the DSSS watermarking technique and then discuss the secrecy of this technique on how to escape detection. An introduction to the basic DSSS principle can be found in Appendix A in [8] (which is available at http://www.cs.uml.edu/~xinwenfu/Infocom09.pdf).

A. Framework of DSSS Watermarking

The framework for the DSSS watermarking technique in [6] is illustrated in Figure 1. The basic idea is: the malicious *interferer* 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: mark generation at the interferer and mark recognition at the sniffer.
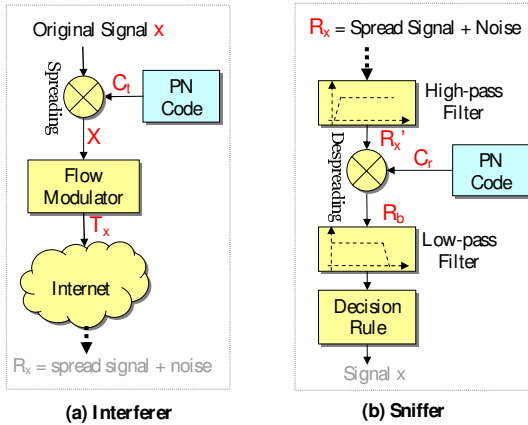

(a) Interferer   **(b) Sniffer**

Fig. 1.   Framework of DSSS-based Traceback

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 traceback [6]. 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 *DSSS 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. Denote 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, 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,

$$R'_x \approx AxC_t + \xi. \tag{4}$$

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,

$$R_b = R'_x \cdot C_r = Ad_tC_t \cdot C_r + \xi \cdot C_r, \tag{5}$$

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, to recognize the original signal x, the DSSS 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 [6]. As indicated in [6], PN code length, original signal length, chip duration t_c, and mark amplitude A all affect traceback performance.

There are mature PN code generators such as *m-sequences code*, *Barker code*, *gold codes* and *Hadamard-Walsh codes* [9], [10] that may be used. Work in [6] used the m-sequence code, which has a sharp autocorrelation function [9]. This characteristic makes it easier for the *sniffer* to accurately synchronize and recognize watermarks in the target traffic.

B. Secrecy of DSSS Watermarking

Secrecy of DSSS watermarks refers to the difficulty of detecting the traceback and it is desired by attackers trying to escape detection. The DSSS-based traceback uses the following mechanisms for secrecy: (i) *Secrecy of the code.* The DSSS 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 traceback. (ii) A carefully chosen mark amplitude A in Formula (2) can be very small in comparison with noise so that the DSSS mark X is covered by 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) *DSSS 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 time and frequency domains.

III. BLIND DETECTION OF DSSS WATERMARKS

In this section, we introduce an approach blindly detecting DSSS watermarks. The detection can be conducted by a sender or a receiver suspicious that she is being maliciously traced. After detection, the sender and receiver may stop communication to thwart such malicious traceback. 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.

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 DSSS 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 DSSS 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 $\vec{x} = \{x_0, \cdots,$

$x_{w-1}\}$ 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 $\vec{C} = \{c_0, \cdots, c_{l-1}\}$, where l is the code length. We assume that bits x_i and x_j ($i \neq j$) are independent, since it is the worst-case of detecting DSSS 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.

Therefore, the modulated signal $\vec{X}$ can be written as follows:

$$\vec{X} = (x_0\vec{C}, x_1\vec{C}, \cdots, x_{w-1}\vec{C}), \qquad (6)$$
$$= (x_0c_0, \cdots, x_0c_{l-1}, \cdots,$$
$$x_{w-1}c_0, \cdots, x_{w-1}c_{l-1}) \qquad (7)$$

where $\vec{X}$ 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, $Pr(x_ic_j = A) = 1/2$, $Pr(x_ic_j = -A) = 1/2$, so that $E(x_ic_j) = 0$, and standard deviation $\sigma = A$. We can use Formula (8) to estimate the autocorrelation of a time series represented by $\vec{X}$,

$$r(\tau) = \frac{1}{(wl - \tau)} \sum_{i=1}^{wl-\tau} y_iy_{i+\tau}, \qquad (8)$$

In (8), τ is the lag and $y_i = x_{\lfloor i/l \rfloor} c_{i\%l}$ is the ith item of $\vec{X}$, 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 (7) and (8),

$$r(kl) = \frac{1}{wl - kl} \sum_{i=0}^{wl-kl-1} y_iy_{i+kl} \qquad (9)$$
$$= \frac{1}{wl - kl}(x_0x_{0+k}\vec{C} \cdot \vec{C}$$
$$+ \cdots + x_{w-1-k}x_{w-1}\vec{C} \cdot \vec{C}), \qquad (10)$$

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

$$r(kl) = \frac{1}{w - k}(x_0x_{0+k} + \cdots + x_{w-1-k}x_{w-1}). \quad (11)$$

$r^2(\tau)$ is the square autocorrelation of spread signal $\vec{X}$ and a time-shifted $\vec{X}$ 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 traceback 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

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

Fig. 2.   Self-similarity of PN 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

Fig. 3.   a Non-Synchronized Window (first row - a traffic segment; second row - the time shifted version of the segment)

(which can be positive or negative). Such results can still be obscured by noise. Squaring such results can guarantee 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 DSSS watermarks.

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

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

The proof of Theorem 1 can be found in Appendix B in [8]. 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 DSSS 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 DSSS-based traceback 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*. The larger 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 DSSS watermarks and permit detection. The detection framework will be presented in details in Section III-C. We give a simple example calculating MSAC in Appendix C in [8].

B. Mean-Square Autocorrelation (MSAC) in a Non-Synchronized Window

In reality, since we do not know the boundary between DSSS 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, as 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 $r^2(\tau)$. 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 DSSS watermarks in a non-synchronized window. Its proof can be found in Appendix D in [8].

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 $q = 1 - p$.

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

where k is an integer.

We have a few observations from Corollary 2:

- The *MSAC* of DSSS 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 DSSS Watermarks

In Section III-A, we demonstrated that the *MSAC* of DSSS watermarks shows periodicity and may be used for detecting malicious DSSS-based traceback. Such detection does not require any traffic synchronization. In this section we present a framework using this feature to detect DSSS watermarks.

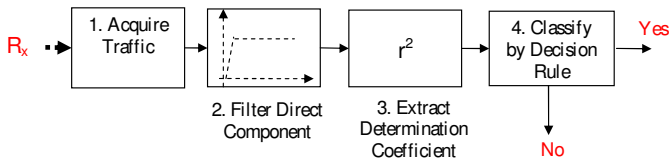Figure 4 shows the four stages of detecting DSSS watermarks via the *MSAC*.

Fig. 4. Workflow of Detecting DSSS Watermarks

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 DSSS watermarks, the acquired traffic segments must be longer than one signal bit. Based on the Nyquist-Shannon sampling theorem [11], 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 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 DSSS watermarks, the unit of τ 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 III-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),

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

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 DSSS watermarked*. Note that periodicity means that peaks of the MSAC appear at regularly spaced intervals.

D. Decision Rule

We now focus on the decision rule for detecting DSSS watermarks. Theorem 1 implies an intuitive decision rule: *if peaks appear at regular intervals, the traffic is DSSS 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 non-synchronized window case. The realistic model of a traffic sample considers both signal components and noise components,

$$y_i = x_i + \xi_i, \quad (15)$$

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_i's autocorrelation. This leads to Theorem 2 for the noise MSAC. Its proof can be found in Appendix E in [8].

Theorem 2: The noise MSAC can be estimated as follows,

$$E(r_\xi^2(\tau)) = \begin{cases} \delta^4(\frac{1}{wl} + 1), & \text{if } \tau = 0, \\ \frac{\delta^4}{wl-\tau}, & \text{if } \tau \neq 0. \end{cases} \quad (16)$$

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 signal-to-noise ratio for the MSAC can be estimated as follows,

$$\frac{E(r_x^2(\tau))}{E(r_\xi^2(\tau))} = \begin{cases} \frac{A^4}{\delta^4}/(\frac{1}{wl} + 1), & \text{if } \tau = 0, \\ l\frac{A^4}{\delta^4}, & \text{if } \tau = kl, 1 \leq k < w, \\ 0, & \text{if } \tau \neq kl, 1 \leq k < w. \end{cases} \quad (17)$$

Proof: Based on $E(r_x^2(\tau))$ in (12) of Theorem 1 and $E(r_\xi^2(\tau))$ 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 $l\frac{A^4}{\delta^4}$ when $\tau = kl$, where $0 < k < w$. Moreover, the longer the code length l, the higher the peak. Therefore, a longer code makes recognition of DSSS watermarks easier.

We introduce an automatic rule in Algorithm 1 for detecting DSSS 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 signal bit. Finally, given a false positive rate, we use hypothesis testing to make decision. If the decision is negative (no DSSS 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 DSSS watermarks are detected, and the *false positive rate*, P_F, as the probability that traffic without DSSS watermarks is misclassified as traffic modulated by a PN code. We can adjust η in Algorithm 1 and derive the corresponding detection

---

Algorithm 1 Detecting DSSS 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 $r_\xi^2(\tau)$, where $\tau \neq klt_a$ and $\tau < (w-1)lt_a$
5:     Calculate the estimated signal MSAC $r_{x,i}^2 = \frac{1}{w-1}\sum_{k=1}^{w-1} r_x^2(klt_a)$ for the ith 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** $result \Leftarrow false$

---

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. Discussion

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 l. The longer the code length, the higher the SNR and the easier detection of DSSS watermarks. This raises a question: can attackers use a shorter code length to make DSSS traceback harder to detect? Unfortunately, this is a dilemma for attackers: as seen in Lemma 1 of [6], achieving reasonable traceback accuracy requires a *longer* code length.

Recall that since our detection relies on the autocorrelation of DSSS watermarks, the DSSS watermarks must contain multiple bits (≥ 2). Another question is: can attackers use just one bit to conduct the traceback to escape detection? The answer is no. When there is just a single bit, the false positive rate of traceback will be too significant at 50% [6]. Given a target false positive rate of 1%, the number of bits must be larger than 7. This will favor the detection of DSSS watermarks again. We will show in Sections IV and V that 7 bits or even fewer is enough for detecting DSSS watermarks.

IV. EVALUATION BY SIMULATIONS

We have theoretically studied the detection of malicious DSSS-based traceback in previous sections. In this section, we use ns-2 simulations to validate our theory of blind detection of DSSS watermarks. The disadvantage of the multi-flow detection approach in [7] is presented in Section IV-D. Results of empirical tests over Tor will be presented in Section V.

A. Simulation Setup

Figure 5 gives the simulation topology. In Figure 5, n_5 and n_7 are Tor-like mixes [12] (no batching or reordering since they are not practical [13]) as used in [6]. 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.
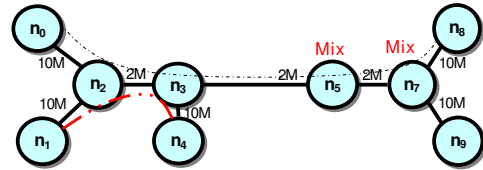

Fig. 5. Topology in ns-2

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 DSSS 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 γ is determined by η 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 γ increases, while 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 DSSS watermarks, we want a high detection rate and a low false positive rate. For the attackers tracing a flow, the ideal scenario is that the false positive rate is as high as the detection rate.
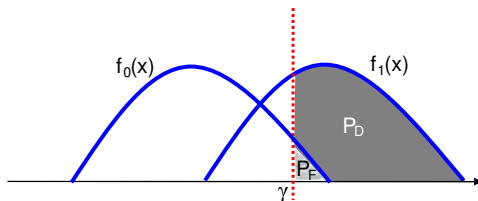
Fig. 6. Calculation of P_D and P_F

C. Detecting DSSS Marks

Now we show the *MSAC* based approach is effective for detecting DSSS 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 III. 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 III-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.

D. Deficiency of Multi-flow Detection in [7]

In [7], a multi-flow approach is proposed to detect interval based watermarks [14], [1] and DSSS-based watermarks [6]. In the case of detecting DSSS 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_{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.

Based on their Formula (6) in [7], we draw P_{t_c} in terms of the number of flows and interval of low traffic rate in Figure 13. The MMPP model used for Figure 13 is trained using ftp downloading sessions over Tor. We can see that the multi-flow detection approach cannot detect a DSSS 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 [7] is that they can detect interval based watermarks [14], [1] given a sufficient number of flows.

V. EXPERIMENTS OVER TOR

To validate our findings, we developed prototype tools and conducted real-world experiments over *Tor* [15], a popular anonymous communication system. Figure 14 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 15 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 [6]. From Figure 15, we can see that the *MSAC* indeed
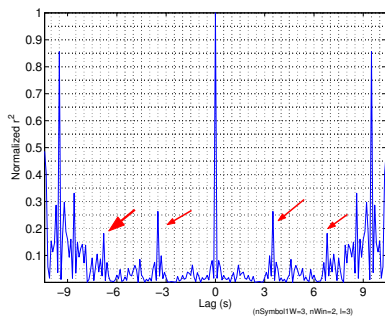
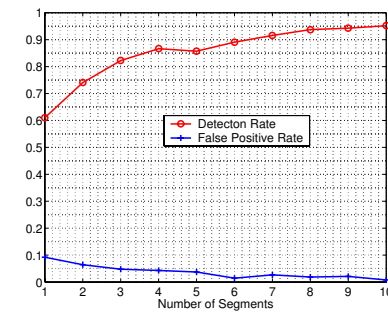Fig. 7. Estimation of Mean-square Autocorrelation

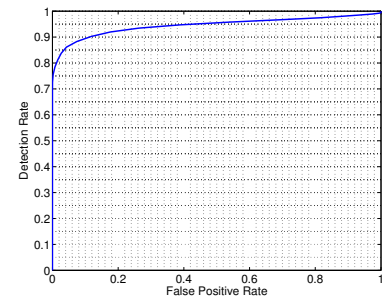Fig. 8. Detection Rate and False Positive Rate v.s. Number of Segments
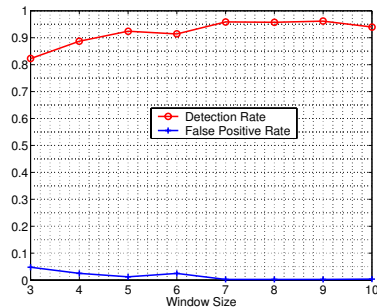

Fig. 9. ROC

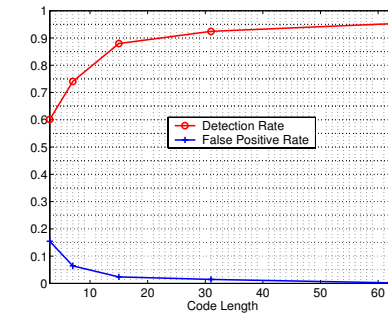Fig. 10. Detection Rate and False Positive Rate v.s. Window Size

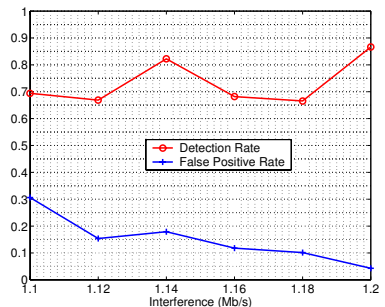Fig. 11. Detection Rate and False Positive Rate v.s. PN code Length

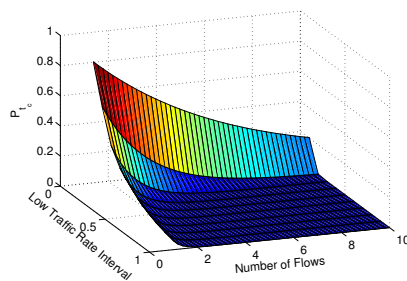Fig. 12. Detection Rate and False Positive Rate v.s. SNR

Fig. 13. Multi-flow Detection in [7] against Malicious DSSS-based Traceback
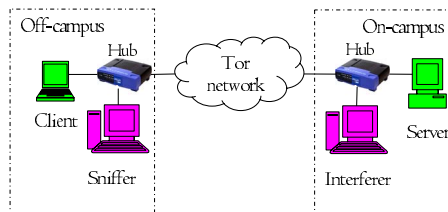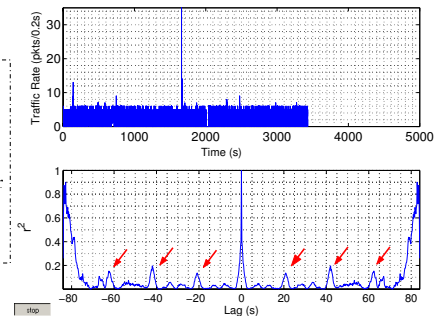

Fig. 14. Experiment Setup

Fig. 15. Detecting DSSS Marks over Tor

demonstrates periodicity at positions of multiple 21 seconds, and we can effectively detect DSSS watermarks.

VI. RELATED WORK

Chaum pioneered the idea of anonymous communication systems in [12]. A good review of various mix systems can be found in [15], [16]. 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 [17] proposed the scheme of using mutual information for the similarity measurement. Levine *et al.* in [18] utilized a cross correlation technique. Murdoch et al. in [3] also investigated the timing based threats on Tor by using some compromised Tor nodes. Fu *et al.* [19] studied a flow marking scheme. Overlier et al. [4] studied a scheme using one compromised mix node to identify the "hidden server"

anonymized by Tor. Yu *et al.* [6] proposed a direct sequence spread spectrum (DSSS) based traceback 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 [20] proposed a scheme that injected nondisplayable content into packets. Wang *et al.* in [21] 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 [22] 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 communication through a mix network with batching strategies that manipulate inter-packet delivery timing, as indicated in [6]. Peng *et al*. in [23] analyzed the secrecy of timing-based watermarking traceback proposed in [21], based on the distribution of traffic timing.

Kiyavash, Houmansadr and Borisov [7] proposed a multi-flow approach detecting the interval-based watermarks (which modify packet timings by selectively delaying some packets [14], [1]) and DSSS watermarks [6]. The approach requires multiple watermarked flows, which may show a unusual long silence period without packets or a 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 apply 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.

VII. CONCLUSION AND FUTURE WORK

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 *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 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 ns2 simulations and experiments on the real world anonymous communication network *Tor* demonstrate a high detection rate with a low false positive rate.

As future work, we will investigate how to detect smarter attacks. For example, an attacker may use multiple PN codes to spread different bits of a signal. To recover the signal, the attacker uses the same sequence of PN codes for despreading. If those PN codes are orthogonal, they cancel each other in Formula (10), and the mean square autocorrelation may not show peaks. This poses a great challenge to detecting the spread spectrum based malicious traceback. We plan a thorough investigation of related issues.

ACKNOWLEDGMENT

REFERENCES

[1] S. Chen X. Wang and S. Jajodia, "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.

[2] Nicholas Hopper, Eugene Vasserman, and David Chan-Tin, "How much anonymity does network latency leak?," in *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS)*, 2007.

[3] S. J. Murdoch and G. Danezis, "Low-cost traffic analysis of tor," in *Proceedings of the IEEE Security and Privacy Symposium (S&P)*, May 2006.

[4] L. Overlier and Paul Syverson, "Locating hidden servers," in *Proceedings of the IEEE Security and Privacy Symposium (S&P)*, May 2006.

[5] Steven J. Murdoch, "Hot or not: Revealing hidden services by their clock skew," in *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS)*, 2006.

[6] W. Yu, X. Fu, S. Graham, D. Xuan, and W. Zhao, "Dsss-based flow marking technique for invisible traceback," in *Proceedings of the 2007 IEEE Symposium on Security and Privacy (S&P)*, May 2007.

[7] Negar Kiyavash, Amir Houmansadr, and Nikita Borisov, "Multi-flow attacks against network flow watermarking schemes," in *Proceedings of USENIX Security*, 2008.

[8] Weijia Jia, Fung Po Tso, Zhen Ling, Xinwen Fu, and Dong Xuan, "Blind detection of spread spectrum flow watermarks," Tech. Rep., University of Massachusetts Lowell, August 2008.

[9] T. F. Wong, "Spread spectrum and code division multiple access," http://wireless.ece.ufl.edu/~twong/notes1.html, August 2000.

[10] ir.J.Meel, "Spread spectrum (ss) - introduction," http://www.sss-mag.com/pdf/Ss_jme_denayer_intro_print.pdf, 1999.

[11] A. V. Oppenheim, A. S. Willsky, and S. H. Nawab, *Signals and systems*, Prentice-Hall, Upper Saddle River, NJ 07458, USA, second edition, 1997.

[12] D. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Communications of the ACM*, vol. 4, no. 2, February 1981.

[13] Xinwen Fu, Wei Yu, Shu Jiang, Steve Graham, and Yong Guan, "Tcp performance in flow-based mix networks: Modeling and analysis," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. TBD, no. TBD, Accepted 2008.

[14] Young June Pyun, Young Hee Park, Xinyuan Wang, Douglas S. Reeves, and Peng Ning, "Tracing traffic through intermediate hosts that repacketize flows," in *Proceedings of IEEE INFOCOM*, May 2007.

[15] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *Proceedings of the 13th USENIX Security Symposium*, August 2004.

[16] G. Danezis, R. Dingledine, and N. Mathewson, "Mixminion: Design of a Type III Anonymous Remailer Protocol," in *Proceedings of the 2003 IEEE Symposium on Security and Privacy (S&P)*, May 2003.

[17] Y. Zhu, X. Fu, B. Graham, R. Bettati, and W. Zhao, "On flow correlation attacks and countermeasures in mix networks," in *Proceedings of Workshop on Privacy Enhancing Technologies (PET)*, May 2004.

[18] B. N. Levine, M. K. Reiter, C. Wang, and M. Wright, "Timing attacks in low-latency mix-based systems," in *Proceedings of Financial Cryptography (FC)*, February 2004.

[19] X. Fu, Y. Zhu, B. Graham, R. Bettati, and W. Zhao, "On flow marking attacks in wireless anonymous communication networks," in *Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS)*, April 2005.

[20] X. Wang, D. S. Reeves, S. F. Wu, and J. Yuill, "Sleepy watermark tracing: An active network-based intrusion response framework," in *Proceedings of 16th International Conference on Information Security (IFIP/Sec)*, June 2001.

[21] X. Wang and D. S. Reeves, "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.

[22] X. Wang, S. Chen, and S. Jajodia, "Tracking anonymous peer-to-peer voip calls on the internet," in *Proceedings of the 12th ACM Conference on Computer Communications Security (CCS)*, November 2005.

[23] P. Peng, P. Ning, and D. S. Reeves, "On the secrecy of timing-based active watermarking trace-back techniques," in *Proceedings of the IEEE Security and Privacy Symposium (S&P)*, May 2006.