DSSS-Based Flow Marking Technique for Invisible Traceback [*]

Wei Yu[†], Xinwen Fu[‡], Steve Graham[‡], Dong Xuan[§], and Wei Zhao[♭]
[†] Texas A&M University, College Station, TX 77843
weiyu@cs.tamu.edu
[‡]Dakota State University, Madison, SD 57042
{xinwen.fu,Steve.Graham}@dsu.edu
[§]The Ohio-State University, Columbus, OH 43210
xuan@cse.ohio-state.edu
[♭]Rensselaer Polytechnic Institute, Troy, NY 12180
zhaow3@rpi.edu

Abstract

Law enforcement agencies need the ability to conduct electronic surveillance to combat crime, terrorism, or other malicious activities exploiting the Internet. However, the proliferation of anonymous communication systems on the Internet has posed significant challenges to providing such traceback capability. In this paper, we develop a new class of flow marking technique for invisible traceback based on Direct Sequence Spread Spectrum (DSSS), utilizing a Pseudo-Noise (PN) code. By interfering with a sender's traffic and marginally varying its rate, an investigator can embed a secret spread spectrum signal into the sender's traffic. The embedded signal is carried along with the traffic from the sender to the receiver, so the investigator can recognize the corresponding communication relationship, tracing the messages despite the use of anonymous networks. The secret PN code makes it difficult for others to detect the presence of such embedded signals, so the traceback, while available to investigators is, effectively invisible. We demonstrate a practical flow marking system which requires no training, and can achieve both high detection and low false positive rates. Using a combination of analytical modeling, simulations, and experiments on Tor *(a popular Internet anonymous communication system), we demonstrate the effectiveness of the DSSS-based flow marking technique.*

1 Introduction

In order to conduct lawful surveillance, law enforcement agencies need the ability to trace Internet communications among those suspected of criminal or terrorist activities. Traditionally, the source and destination IP addresses in an IP header have allowed investigators to trace communication sessions and determine corresponding participants, timing, frequency, and quantity. However, the proliferation of anonymous communication systems [1, 2, 3, 4] on the Internet has posed significant challenges to effectively tracing communications. For example, web file downloading can be disguised using anonymous communication systems such as *Tor* [4, 5], preventing detection of illegal use in cases, such as child pornography [5]. Terrorists or criminals might use anonymous communication systems to exchange information and develop plots, without being detected.

To preserve the capacity of tracing Internet communications despite anonymous channels, we must use traffic characteristics other than easily modified IP header information. For this purpose, we may use *flow marking*, introduced in [6]. To determine whether a sender is communicating with a receiver, an investigator, known as the *interferer*, can embed a series of marks (signals with a specific pattern) into the sender's traffic by interfering with the sender's outbound messages. Another investigator, known as the *sniffer*, eavesdrops on the receiver's inbound traffic. If a similar pattern of embedded marks is found in the receiver's traffic, the investigators know that the sender is communicating with the receiver. By tracing the marks, investigators may construct the full communication path.

[*]This work was partially sponsored by South Dakota Governor Individual Research Seed Grant Program, and the Project of The South Dakota Electronic Health Record Assessment (SDEHRA) from South Dakota Department of Health. Any opinions, findings, conclusions, and/or recommendations expressed in this material, either expressed or implied, are those of the authors and do not necessarily reflect the views of the sponsors listed above. The authors would like to acknowledge Ms. Larisa Archer for her dedicated help to improve the paper and Dr. Tom Halverson for his support of this project.

There are two requirements for successfully tracing anonymous communications: accuracy of the traceback and invisibility of the traceback. While accuracy has received much attention, invisibility has been largely ignored, although it is also a vital requirement. When traceback attempts are identified, the correspondents will simply stop their communications, evading further detection. The correspondents may even develop countermeasures to fool or mislead investigators (e.g., by abusing some targets and feigning communications). To the best of our knowledge, existing techniques have failed to meet both these requirements simultaneously [6, 7]. For example, marks in a periodic pattern such as [6] are easy to introduce, since an investigator may just interfere with a target traffic flow periodically. However, a periodic pattern of marks may result in a high false positive rate, since such traffic markings may introduce a similar periodic pattern into other traffic sharing a link. Furthermore, when a *Fourier* Transform is applied to a traffic flow containing a periodic pattern, the periodic pattern emerges as obvious in the frequency domain to investigators as well as the correspondents, who may take some countermeasures to defeat this approach. Hence, a successful flow marking must effectively trace the communications while remaining undetectable by anyone other than investigators.

In this paper, we develop a novel class of flow marking technique for invisible traceback based on *Direct Sequence Spread Spectrum* (DSSS). In this technique, the investigator introduces invisible DSSS marks into a target traffic flow. The marks correspond to a signal modulated by a *Pseudo-noise* (PN) code, known only to the investigators. Only those knowing the code can correctly recover the original signal and identify the communication relationship. The PN code modulated signal will appear as innocent noise in both the time and frequency domains, so it is difficult for others to detect the presence of such signals in the target traffic[1]. Therefore, using a DSSS-based technique, we are able to trace anonymous communication while evading detection.

We develop a new DSSS mark generator that embeds a secret spread signal using a secret PN code into a target flow at the transmitter. To recover the signal at the receiver, we use digital filters to remove direct current components, which correspond to network traffic seasonal variations, and high frequency noise from target traffic flow, so we can effectively recover the DSSS marks. Our DSSS-based technique has a simple and effective decision rule, compared with other threshold-based techniques that normally require lengthy and impractical training processes [8, 9]. Using a model, we derive formulas for detection and false positive rates for our traceback technique. We discuss how to determine various parameters such as an appropriate PN code length. We also address practical issues such as PN code synchronization and tracing multiple traffic flows simultaneously.

Besides theoretical analysis, we have also conducted extensive evaluations to our proposed approach using simulations and real-world experiments. We use *ns-2* simulations to explore the effectiveness of our DSSS-based technique. We show that even with low signal to noise ratios, our technique is robust and able to correlate sender and receiver communication relationships at a probability of 100%, when the PN code length is reasonable large. We show that the false positive rate is also significantly suppressed. Our data show that the DSSS-based technique is capable of effectively invisible traceback, since there is no clear difference between the traffic with marks and the traffic without marks in either the time domain or the frequency domain. We show that our technique is effective for tracing multiple traffic flows simultaneously, using different low cross-correlated PN codes. We also show that when anonymous communications systems (mixes) adopt sophisticated batching strategies, our technique can still achieve a high detection rate.

We developed a suite tools and performed a set of real-world Internet experiments on communications using *Tor*, a popular anonymous communication network for transporting TCP streams over the Internet. Our data validate the theoretical and simulation findings and demonstrate that our DSSS-based technique can invisibly track anonymous traffic flows using *Tor*, even when such flows exhibit wild dynamics. To the best of our knowledge, there are few other efforts applying DSSS techniques for tracing communications.

The rest of the paper is organized as follows. We introduce mix networks, DSSS-based techniques, and flow marking in Section 2. We present the detailed design of the DSSS-based flow marking technique in Section 3. In Section 4, we conduct the theoretical analysis of detection and false positive rates, and discuss how to determine parameters. In Section 5, we use *ns* simulations to demonstrate the effectiveness of DSSS-based flow marking for traceback. In Section 6, we present our experiments using *Tor*, which validate our findings. We review related work in Section 7. We conclude this paper in Section 8.

2 Background

In this section, we first give an overview of *mix networks*, and then introduce the basic *Direct Sequence Spread Spectrum* (DSSS) technique followed by basics of *flow marking*.

[1] Because of this property, the correspondents cannot use the approach in [6], which relies on identifying periodic patterns in traffic, to remove embedded signals discussed in this paper.

2.1 Mix Network

Mix networks have been used by popular anonymous communication systems such as *Tor*. In a mix network, senders route their packets through a series of mixes. A mix manipulates packet delivery by batching, reordering, and forwarding the packets in order to prevent traffic analysis from correlating input packet and output packets, and degrading anonymity. *Tor* does not use batching and reordering because of quality of service concerns.

Work in [10] provides a relatively complete list of batching strategies for mix networks. These strategies can be applied by mix networks for message-level (packet-level) applications such as *remailer* [11]. But not all of them are appropriate for mix networks used for flow-based anonymity applications. For example, in a threshold mix, a mix can transmit the batch of packets only if the number of collected packets exceeds a pre-defined threshold. This may cause serious problems for TCP traffic flows. For example, if the first (SYN) packet cannot be exchanged between a sender and receiver, the TCP flow cannot start and communication through the mix network fails. We use three batching strategies for mix networks and flow-based applications in Table 1 [6].

2.2 Direct Sequence Spread Spectrum Technique

Spread spectrum (SS) is a transmission technique in which a pseudo-noise (PN) code, independent of the original data signal, is employed to "spread" the signal over a bandwidth greater than the original data signal bandwidth. At the receiver, the signal is "despread" using a synchronized copy of the PN code. Spread spectrum techniques are often used in code division multiple access (CDMA) systems. There are three classes of spread spectrum techniques: direct sequence spread spectrum (DSSS), frequency hop spread spectrum (FHSS), and time hop spread spectrum (THSS) [12]. Any of these techniques could be applied in our approach. In this paper, we adopt DSSS because of its easy implementation and wide application [12]. The DSSS technology was initially used in military communication systems to provide anti-jamming and secured communication [13]. In wireless communication, DSSS technology has been widely used to improve the communication efficiency [14]. In addition, DSSS technology has other broad applications [15, 16, 17].

Figure 1 shows the basic principle of DSSS. The original signal d_t at the transmitter is a series of binary symbols (here we use bits encoded as $+1$ or -1 instead of 1 or 0), although the signal could be encoded by other schemes such as QPSK (Quadrature Phase Shift Keying) [18]. The *symbol duration* for both symbol $+1$ and -1 is T_s seconds, so the symbol rate $R_s = 1/T_s$. A PN code c_t of $+1$ and -1 is generated at the transmitter and shared between the receiver. Each bit (denoted as *chip*) in the PN code lasts for T_c seconds (denoted as *chip duration*), so the chip rate is $R_c = 1/T_c$. N_c is the number of chips per symbol and is also called as the *PN code length*. These concepts are illustrated in Figure 2.

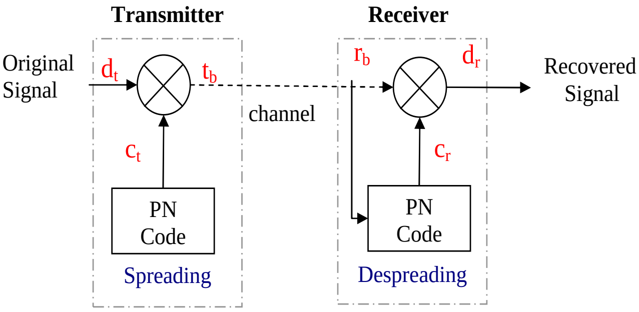

Figure 1. Spreading and Despreading in DSSS

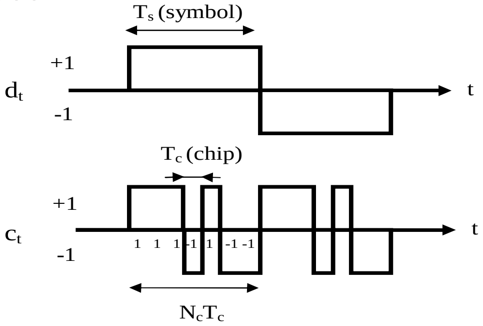

Figure 2. Concepts in DSSS

Now we discuss the spreading process at the transmitter. Without loss of generality, we discuss using a PN code to spread an original signal d_t of one bit, $+1$ or -1. d_t is directly multiplied with the PN code c_t, which is independent of the signal, to produce the transmitted signal $t_b = d_t c_t$, where c_t is a $1 \times N_c$ vector with elements corresponding to the chip values, either $+1$ or -1 drawn from the PN code at the transmitter.

The transmitted signal t_b passes through the communication channel and reaches the receiver. If there is no interference along the channel, the received baseband signal $r_b = t_b = d_t c_t$. To recover the original signal from r_b, r_b is multiplied with a $1 \times N_c$ subsequence from the PN code called c_r at the receiver. We have the recovered signal

$$d_r = \frac{\sum (r_b \cdot c_r)}{N_c} = d_t \frac{\sum (c_t \cdot c_r)}{N_c}, \qquad (1)$$

where the operator of $\cdot$ refers to direct multiplication of vectors and the operator of $\sum$ adds up all the elements of a vector. There are two cases here:

Table 1. Batching Strategies

Strategy Index	*Name*	*Adjustable Parameters*	*Algorithm*
S_0	Simple Proxy	*none*	No batching (or reordering).
S_1	Timed Mix	$< t >$	If timer with period t expires, send all the packets queued in the last interval.
S_2	Stop-and-go (SG) Mix (or Continuous Mix)	$< \mu, \sigma^2 >$	Each packet is assigned a delay (deadline) satisfying a distribution with mean μ and variance σ^2. A packet is sent out when its deadline is reached.

1. $c_r = c_t$: If the PN code at the receiver is synchronized to the PN code at the transmitter, $c_t \cdot c_r = \mathbf{1}$, where $\mathbf{1}$ refers to a $1 \times N_c$ vector with all elements equal to 1. The original signal d_t can be recovered by $d_r = d_t \frac{\sum(c_t \cdot c_r)}{N_c} = d_t \frac{N_c}{N_c} = d_t$.

2. $c_r \neq c_t$: If the receiver does not have the right PN code, $\sum(c_t \cdot c_r)/N_c \neq 1$ and $d_r \neq d_t$. So a receiver without the PN code cannot reproduce the original signal d_t.

2.3 Flow Marking

Flow marking is a general technique for tracing Internet communications despite anonymous channels [6]. Figure 3 illustrates the basic idea of flow marking. Alice is communicating with Bob through a mix network. To determine if Alice is communicating with Bob, an *interferer* can embed a pattern of *marks* into Alice's messages by interfering with her outbound traffic. A *sniffer* eavesdrops on Bob's inbound traffic. If a similar pattern of marks is discovered in Bob's traffic, the *sniffer* knows that Alice is communicating with Bob. By tracing marks link-by-link, we can reconstruct the path between Alice and Bob.

In this paper, we explore the dynamics of flows (caused by flow-control) and interfere with the flows, marking them for traceback. We focus on TCP flows for our DSSS-based flow marking technique because of TCP's dominant role in Internet traffic, e.g., TCP flows constitute 60% $\sim$ 90% of the Internet traffic [19, 20].

3 DSSS-Based Flow Marking Technique

In this section, we investigate how to apply DSSS to mark traffic flows of interest. Since a new category of modulation and demodulation for flow marking context is introduced, we need to develop a new mark generation and recognition processes. We introduce the process of DSSS-based flow marking, and then we present the detailed design of key components followed by some extension.

3.1 Workflow

Generally speaking, there are two important modules (mark generation and recognition) in the DSSS-based flow marking system. Figure 4 describes the basic processes of these two modules.

Mark generation module at the transmitter:

1. An original signal d_t of "+1" or "-1" is to be transmitted (to transmit n-bits of signal, just repeat the following steps). Similar to the process in Figure 1, we obtain the transmitted baseband signal t_b as

$$t_b = d_t c_t, \tag{2}$$

where c_t is a PN code with chip duration T_c.

2. t_b is then used to modulate a target traffic flow. When a chip is $+1$, *weak interference* is applied against the flow so that the flow has a high rate for T_c seconds. When a chip is -1, *strong interference* is applied against the flow so that the flow has a low rate for T_c seconds. If we assume that the flow has an average traffic rate of D, then the high rate is $D + A$ and the low rate is $D - A$, where A is denoted as *mark amplitude*. The rate of the target traffic flow should be large enough for investigators to introduce the marks by interference without obvious effects on the target traffic flow. Therefore, the transmitted signal tx can be represented by,

$$tx = A d_t c_t + D. \tag{3}$$

3. The modulated flow is carried through the Internet, where there exists noise created by cross traffic and intentional interference. We treat all noise as an aggregated factor.

Mark recognition module at the receiver:

1. Denote noise as a random variable w. We can formulate the received signal rx as

$$rx = A d_t c_t + D + w. \tag{4}$$

The *sniffer* in Figure 3 derives rx by capturing the traffic at the receiver, then dividing it into segments. Each segment

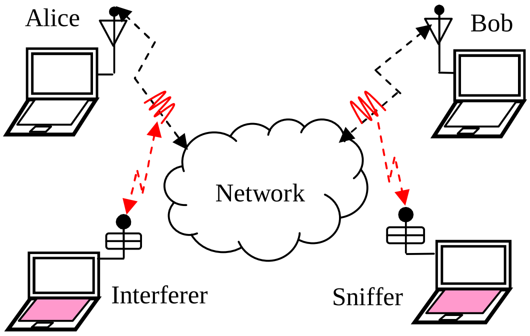

Figure 3. Flow Marking

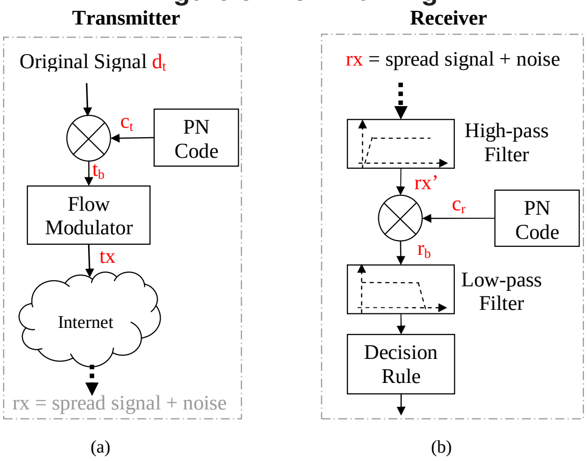

Figure 4. Mark Generation and Recognition Modules

lasts for a chip duration of T_c seconds, and the average traffic rate of each segment can then be calculated. Average rates for N_c continuous segments constitute rx. Recall that all items in (4) are $1 \times N_c$ vectors and N_c is the PN code length per symbol and the number of chips for 1-bit of the original signal.

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

$$rx' \approx Ad_t c_t + w. \tag{5}$$

3. A locally generated PN code c_r, the same as the code at the transmitter, is used to despread the filtered received signal rx' to derive the received baseband signal r_b,

$$r_b = Ad_t c_t \cdot c_r + w \cdot c_r. \tag{6}$$

4. A low-pass filter is used to filter out high frequency noise. We then use a decision rule to classify the received signal as $+1$ or -1 of length n.

3.2 Key Components

We have presented the process of the DSSS-based flow marking system. In the following, we will present our detailed design of key components such as DSSS marks, filters, and decision rule.

3.2.1. DSSS Marks. As we mentioned in Section 1, invisibility (the difficulty of detecting the tracing by anyone other than investigators) is one important goal we want to achieve with our technique. Here we develop details of using DSSS marks. The major advantage of DSSS marks is that they are invisible without knowledge of the PN code. Since the sender and receiver don't know the PN code, it is very difficult for them to recognize the existence of marks embedded in their traffic flow. Only investigators introducing and recovering the DSSS marks (*interferer* and *sniffer*) can recognize them. In addition, a good PN code and the corresponding modulated signal appear random for anyone who doesn't know the code[2]. The spectrum of the modulated signal in the frequency domain resembles random noise. In general, the longer the code length, the harder it is to detect.

A carefully chosen mark amplitude A in (3) can be very small so that the transmitted signal tx is covered by the noise w in the received signal rx. The recognition process will effectively restore the spread signal to its narrow band and recover the original signal d_t from the noise (even though the mark amplitude is small). Other parameters such as PN code length N_c, original signal length n, and chip duration T_s impact how well flow marking can be performed. Detailed analysis of the impact and determination of these parameters is further addressed in Section 4.

There are mature PN code generators such as *m-sequences code*, *Barker code*, *gold codes* and *Hadamard-Walsh codes* [18, 21] that we may adopt. In this paper, we use the m-sequence code, which has the best autocorrelation (it only highly correlates to itself with a sharp autocorrelation peak) [18]. The improved autocorrelation makes it easier for the *sniffer* to accurately synchronize and recognize the pattern embedded in the marked flow. The source code for generating the m-sequence code is available at [22].

In addition, in order to embed marks into traffic such as TCP flows, an *interferer* may exploit the dynamics of flow-control and use efficient denial-of-service approaches to modulate the signal [23].

3.2.2. Digital Filters. Since the target traffic flow of interest with embedded spread signal is carried through the Internet, where there exists noise created by cross traffic and intentional interference, the ability to effectively recognize marks in the presence of noise is a critical issue for DSSS-

[2] The original signal is also designed to appear random in order to maintain the invisibility of the traceback.

based flow marking. To address this issue, we develop digital filters (a high-pass filter followed by a low-pass filter) to process the received signal rx and recover the original signal d_t.

As shown in Section 3.1, the high-pass filter is used to remove the direct current component D of the received signal in (4). Therefore it decouples the embedded marks and the target traffic flow. After despreading the output of the high-pass filter, the low-pass filter is applied to remove interference of high frequency noise. With these procedures, the original signal d_t can be effectively recovered.

3.2.3. Decision Rule. Since the digital filters can effectively recover the original signal d_t from the target traffic flow, this also leads to a simple and effective decision rule without the training process. We obtain the decision rule based on results presented in Theorem 1. The detailed proof of this Theorem is available in Appendix A of [24].

Theorem 1 *Denote rx_{+1} as the result of $\sum r_b/N_c$ when the original signal d_t is $+1$ and rx_{-1} as the result of $\sum r_b/N_c$ when the original signal d_t is -1. When the received signal passes the low-pass filter, we have*

$$rx_{+1} = \sum r_b|_{d_t=+1}/N_c = A(1-1/N_c^2), \quad (7)$$

$$rx_{-1} = \sum r_b|_{d_t=-1}/N_c = -A(1-1/N_c^2), \quad (8)$$

$$rx_{+1} = -rx_{-1}, \quad (9)$$

where A is the mark amplitude and N_c is the PN code length.

Since $1/N_c^2 < 1$, $rx_{+1} > 0$ and $rx_{-1} < 0$. As we mentioned, the low-pass filter is used to remove high frequency noise, since the signal bandwidth is $[-R_s, R_s]$, where $R_s = 1/T_s$. Given this result, it is easy to derive a decision rule as

$$d_r = \begin{cases} +1 & ,\sum r_b/N_c \geq 0 \\ -1 & ,\sum r_b/N_c < 0. \end{cases} \quad (10)$$

From (10), we can see that the decision rule used to recognize an original signal d_t at the receiver is very simple and it does not need any tedious or impractical training. This simple decision rule is an advantage of our technique compared with many existing approaches [8, 9].

3.3 Extension

3.3.1. Synchronization of DSSS Marks. For proper operation in recognizing the original signal d_t, the DSSS-based flow marking requires that the locally generated PN code c_r at the receiver be synchronized with the PN code c_t at the transmitter in both frequency and time. Synchronizing the frequency is easy, since the *interferer* and *sniffer* can share the same values of parameters such as the chip duration T_c and code length N_c. Here we focus on the time synchronization. Although a PN code has a sharp peak in its autocorrelation function, a misalignment of c_r against c_t may not recover the original signal. In our case, this misalignment comes from the uncertainty of propagation delay between the transmitter and receiver over the Internet. Figure 5 shows a misaligned c_r against c_t, i.e, the PN code generated at the transmitter and the one at the receiver have phase difference $\triangle t$.

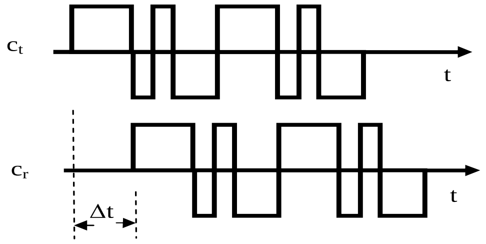

Figure 5. Misaligned c_r against c_t

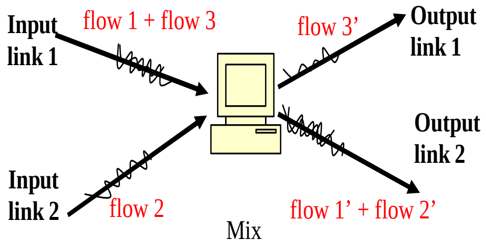

Figure 6. Tracing Multiple Flows Simultaneously

To address this problem, we develop a matched filter based approach for PN code synchronization. A matched filter calculates the correlation function of the filtered received signal rx' in (5) and the locally generated PN code c_r at a *sample interval* T_s. This can be performed on-line and gives the shortest acquisition time. In our case, the signal detection process has only soft real time requirements. We can dump the traffic first and then conduct the signal recovery via high performance computers. In this way, in order to have an accurate synchronization between the sender's PN code and the receiver's PN code, we need to use an appropriately small sample interval. Here, the matched filter based approach uses a sliding window iteratively moving back and forth, therefore capturing a segment of traffic data for the best signal match. As such, we can determine whether the expected signal exists in the segment of dumped traffic.

3.3.2. Tracing Multiple Flows. In order to achieve efficient traceback, it is better if we can trace multiple flows in parallel. However, the different flows may go through the same mix and interfere with each other. As shown in Figure

6, there are three flows (*flow 1*, *flow 2* and *flow 3*) entering a mix. The *flow 1* and *flow 2* are integrated in the output link 2 of the mix.

To address this problem, we assign different PN codes to different target traffic flows of interest. Since the generated PN codes are of low cross-correlation, this makes the flows, embedded with signals modulated by different PN codes, cause little interference to each other. Thus, it is feasible that DSSS-based flow marking technique can trace multiple flows in parallel. Using Figure 6 as an example, we can use PN code one, $c_{t,1}$, to mark the flow 1 on input link 1 and PN code two, $c_{t,2}$, to mark the flow 2 on input link 2. On output link 2, a *sniffer* using copies of these two PN codes, can apply the same techniques discussed previously to identify each flow. The detailed analysis of tracing multiple flows is shown in Appendix B of [24].

4 Analysis

We have presented the DSSS-based flow marking technique. In this section, we study the effectiveness of DSSS-based flow marking and derive formulas for detection rate and false positive rate of DSSS-based flow marking. Based on these results, we also discuss how to determine important parameters such as PN code length N_c.

4.1 Robustness in the Presence of Noise

Internet traffic introduces noise, interfering with the spread signal t_b embedded in the target traffic flow. Now we investigate how noise influences the recovery of a 1-bit original signal. The results are presented in Lemma 1. The detailed proof of this Lemma is available in Appendix C of [24].

Lemma 1 *Assume that the noise is a Gaussian white noise (WGN) process with distribution $N(0, \sigma_w^2)$. Detection rate P_D is defined as the probability that a 1-bit original signal is correctly recognized. We have*

$$P_D = 1 - \frac{1}{2} erfc(\sqrt{\varepsilon}), \tag{11}$$

where $erfc(.)$ is the complementary error function (monotonically decreases with ε), $erfc(z) = \frac{2}{\sqrt{\pi}} \int_z^\infty e^{-t^2} dt$, and ε is represented in (12)

$$\varepsilon = \frac{(N_c^2 - 1)^2}{2N_c^3} \frac{A^2}{\sigma_w^2}. \tag{12}$$

In engineering, the term of A^2/σ_w^2 in (12) is denoted as signal to noise ratio (*SNR*), which quantifies the marking interference strength over the noise.

Given P_D as the detection rate of the 1-bit original signal in Lemma 1, we have the following theorem for the detection rate of an n-bit original signal d_t.

Theorem 2 *Detection rate $P_{D,n}$ for detecting an n-bit original signal d_t is*

$$P_{D,n} = P_D^n. \tag{13}$$

We have the following observations from Lemma 1 and Theorem 2: Detection rate $P_{D,n}$ increases with SNR. In the context of communication surveillance, we cannot have a large SNR, which would make the marking pattern prominent in the marked traffic flow and raise suspicions of traceback attempts. Detection rate $P_{D,n}$ of n-bit original signal d_t also increases with the increasing PN code length N_c. This helps us to achieve a decent detection rate with a small SNR.

4.2 False Positives in the Case of Zero Signals

We have discussed how to detect an original signal under the assumption that there are spread signals embedded into a target traffic flow (sender and receiver are communicating). In the case that there is no spread signal embedded into the target traffic flow (sender and receiver are actually not communicating), there is the possibility that the decision rule (10) makes a wrong decision (false positive).

As shown in Corollary 1, the classifier (10) has equal probability of producing a “1” or “-1” when there is no spread signal embedded in the target traffic. The detailed proof of this Corollary is available in Appendix D of [24].

Corollary 1 *Detection rate P_D and error rate P_e (probability for mistakenly recognizing the signal that does not exist) for 1-bit original signal approach 50% when signal noise ratio SNR approaches 0 (the case that there is no signal).*

Then the false positive rate $P_{F,n}$ for recognizing n-bit original signal will be

$$P_{F,n} = \frac{1}{2^n}. \tag{14}$$

As we can see, we will have lower false positive rates, as the original signal length n gets longer. In reality, investigators know when a signal is embedded into the target traffic flow. So the embedded signal should appear within a very narrow interval at the receiver. In this way, the actual false positive rate is reduced and roughly equal to that in (14).

4.3 Suppressing False Positives Caused by Cross Flows

When the target traffic flow of interest is interfered with embed the spread signal, cross traffic will also be influenced when it shares some link with the target traffic flow. This will cause possible false positives such that the *sniffer* incorrectly identifies the communication (e.g., in case of Alice actually communicating with Bob, the investigators mistakenly determine that Alice is also communicating with John).

However, our DSSS-based technique can effectively suppress this kind of false positive. Basically, when the target traffic flow rate decreases, the cross traffic flow rate increases. When the target traffic flow rate increases, the cross traffic flow rate decreases. That is, in an ideal case, a spread signal $tx_c = -tx$ is embedded into the cross traffic flows. By using a derivation similar to Theorem 1, we can have the despread signal for a cross traffic flow as

$$r_b = -d_t(1 - 1/N_c^2). \qquad (15)$$

Therefore, after despreading, we have

$$\begin{aligned} rx_{+1} &= -A(1 - 1/N_c^2), &(16)\\ rx_{-1} &= A(1 - 1/N_c^2). &(17) \end{aligned}$$

As we can see, the recovered signal is simply the inverse of the original signal embedded into the target traffic flow. Since the recovered signal does not match the original signal, false positive caused by cross flows can be effectively suppressed. As we mentioned, other approaches have a problem with high false positive rate caused by cross flows [6].

4.4 Determination of DSSS-Based Flow Marking Parameters

We can use above analytical results to determine DSSS-based flow marking parameters. We give the detail of parameter determination as follows. 1) *Original Signal Length n:* Given the false positive rate $P_{F,n}$ in (14), we can determine the original signal length n. For example, given the false positive rate of 1.5% (or 0.7%), we can use the original signal of length 6 (or 7). 2) *Mark Amplitude A:* From Lemma 1 and Theorem 2, we can see that a bigger *SNR* renders a high detection rate. However, since an invisible mark is desired, the mark amplitude A should be chosen as small as possible so that the marks can be covered by noise and dynamics of traffic flow. 3) *Code Length N_c:* Given the detection rate $P_{D,n}$, original signal length n and mark amplitude A, we can further determine the code length N_c by resolving (11) and (13). 4) *Chip duration T_c:* In practice, the duration of a traffic flow may be limited. To embed enough marks into the target traffic flow, the chip duration should be as small as possible. As the mark is introduced by interference, it takes time for the interference to take effect (e.g., it takes time for flow-control to react [23]). A too small T_c may not provide enough time for the traffic flow to reach a necessary traffic rate given the intensity of interference. The choice of chip duration T_c has a direct impact on mark amplitude, and thus *SNR*. The simulations in Section 5 and experiments on *Tor* in Section 6 will further validate these findings.

5 Evaluation by Simulations

We have systematically developed the DSSS-based flow marking technique for traceback in previous sections. In this section, we use *ns-2* simulations to investigate the effectiveness of this technique. We use TCP traffic as the example because of TCP's dominant role in Internet traffic.

5.1 Simulation Setup

Figure 7 gives the simulation topology. We conduct two types of simulations: 1) tracing a single target flow and 2) tracing multiple flows in parallel. Except where it is explicitly stated, we focus on the single target flow case in our discussion. In Figure 7, n_5 and n_7 are mixes, which may use batching schemes, illustrated in Table 1. The target FTP flow runs from node n_0 to node n_8 throughout the simulations. There are also cross FTP 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.

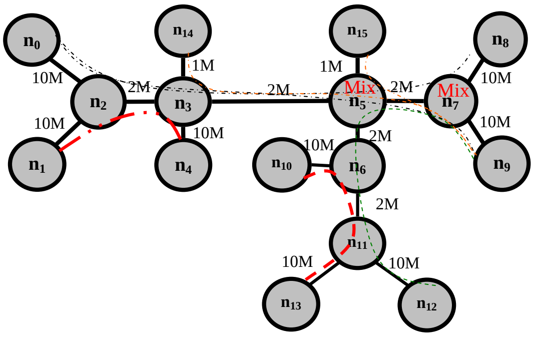

Figure 7. Topology in ns-2 Simulations

In our simulation, the CBR 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, we mark the target FTP flow by adjusting its rate through the interference of the CBR traffic. To use the DSSS technique and recover the original signal, we have to obtain a time series of the FTP flow rate. We use a sample interval of

0.1s (the sampling rate is 10Hz). In order to recover the spread signal, the sampling period should be less than half of the chip duration based on the Nyquist sampling theory [25]. In other words, our sample interval of 0.1s is suitable for a chip duration greater than or equal to 0.2s. We can increase the sampling rate to increase the quality of the recovered signal. This also increases the accuracy of PN code synchronization.

In practice, we cannot accurately predict the delay between a sender and receiver, because of the complexity and dynamics of the Internet. In our simulations, in order to synchronize the PN code at the receiver with that of the sender, we use a coarse estimation of the delay, which is roughly equal to the cumulative propagation and queueing delay of the flow path. Then, we use the matched filter based approach discussed in Section 3.3.1 to search for the best match within a certain search range. We set that range as $[-0.5s, 0.5s]$ in our simulation cases.

5.2 Mix Implementation in ns-2

We have implemented mix batching techniques as in Table 1 in *ns-2*. A mix is implemented as an *ns-2* node that is placed between sender and receiver nodes. Packets entering a mix can be batched and reordered based on the mix type: MixBoxSG (continuous-time/stop-and-go mix) and MixBoxT (timed mix). A simple mix proxy behaves like a general router, except that all packets passing through it have the same size.

Node/MixBoxSG consists of a Classifier/MixBoxSG that sits in front of the default classifier, which first delays and reorders packets, and then sends packets. When a packet arrives at the mix, a deadline is generated for the packet. Packets are buffered and sorted so that the packet with the earliest deadline is stored at the beginning of the buffer. A timer event is generated for the first packet in the buffer. If a new packet's deadline is earlier than the deadline of the first packet already in the buffer, we cancel the current timer event and schedule a new event for the new packet. When a packet is sent, Classifier/MixBoxSG passes the packet on to the default classifier.

Node/MixBoxT also consists of a Classifier/MixBoxT that sits in front of the default classifier, delays and reorders packets, and sends packets in batches at fixed intervals. Packets are stored in a buffer. The packet buffer is implemented as a *multi-map*, in which the key is a generated value from a uniform distribution. A timer expires periodically, flushing out packets from the buffer. When a packet is sent, Classifier/MixBoxT passes the packet on to the default classifier.

5.3 Effectiveness of DSSS-Based Flow Marking

For the following experiments, unless it is explicitly stated, the detection rate refers to the probability that an n-bit signal is correctly recognized, as we analyzed in Section 4. Figure 8 shows the detection rate of a 7-bit signal, $\{$ *1 -1 1 1 -1 1 -1*$\}$ in terms of the code length for different CBR interference rates. In this set of simulations, the chip duration is fixed at 0.4 seconds.

From Figure 8, we make the following observations. The DSSS-based flow marking technique can correlate a sender and receiver at a probability of 100% for all CBR traffic rates when the PN code length is reasonably large. For example, when the CBR traffic rate is 1.118Mbps, a 15-chip PN code achieves the detection rate of 100%. This validates the effectiveness of the DSSS flow marking technique. As predicted in Lemma 1 and Theorem 2, the detection rate increases with the PN code length.

Figure 9 shows the detection rate in terms of the CBR traffic rate when the PN code length is 7. We can see that when the CBR traffic rate increases, the detection rate also increases. This is because when a CBR traffic with a higher rate is on, the FTP flow gets a smaller share of the link bandwidth from n_2 to n_3. This increases the mark amplitude A in (12), and yields a greater SNR. The greater SNR leads to a better detection rate.

Figure 10 shows the detection rate in terms of the chip duration. We can see that when the CBR traffic rate is either 1.116Mbps or 1.118Mbps, the detection rate increases with the increasing chip duration until it reaches 100%. Again, this comes from TCP's flow-control dynamics. When the CBR traffic is turned on, it takes time for the FTP flow to reach its lowest rate. When the CBR traffic is turned off, it also takes time for the FTP flow to return to its previous rate. So, a longer chip duration increases the mark amplitude A in (12), and yields a greater SNR, and a better detection rate. A longer chip duration also reduces the error introduced by PN code synchronization, again contributing to a better detection rate.

5.4 Invisibility of DSSS-Based Flow Marking

Invisibility (the difficulty of detecting the tracing by anyone other than investigators) is another goal we hope to achieve with our technique. To demonstrate this capability, Figure 11 shows the average rate for FTP traffic with marks and without marks and the corresponding frequency domain spectrum of such traffic[3]. Figure 12 shows the Probability Density Function (PDF) of traffic rates with marks and

[3] We obtain the FFT of the traffic rate time series and then derive its power spectrum.

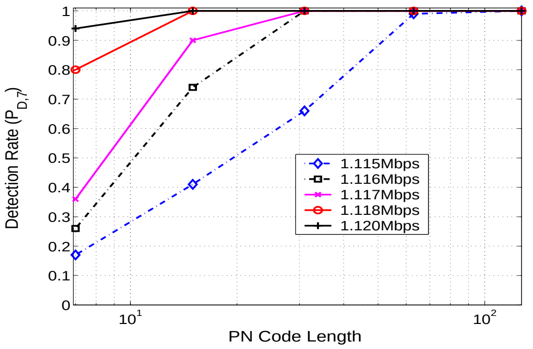

Figure 8. Detection Rate v.s. PN Code Length

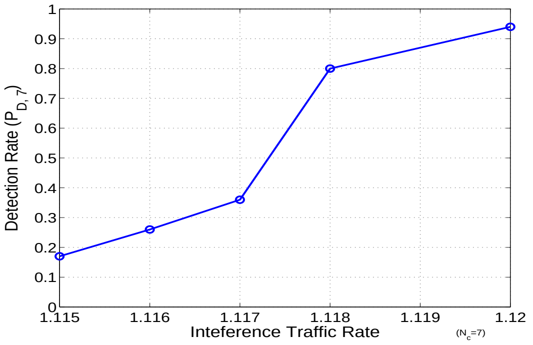

Figure 9. Detection Rate v.s. Interference Traffic Rate

without marks in both time and frequency domains. The CBR traffic has a rate of 1.116Mbps, the PN code length is 7, the signal length is 7 and the chip duration is 0.4 seconds. Recall that we sample the traffic at 0.1 seconds, that is, we divide the traffic into segments, with each lasting for 0.1 seconds, after which the average rate for each segment is calculated. It can be observed that in the time domain, there is no clear difference between traffic with marks and traffic without marks. We also observe that there is no apparent difference between the power spectrum of traffic with marks and without marks.

5.5 False Positive Rate

Figure 13 shows the false positive rate when we try to recognize a signal from traffic where no marks exist (sender and receiver are not communicating). In our simulation, we vary signal length from 1 to 7, and for each fixed signal length we measure the false positive rates for codes of different lengths (from 2 to 7). The false positive rate for each signal length is calculated as the average of the "detection rate" for the different code lengths tested with that signal. From Figure 13, we can see that the false positive rate decreases exponentially with the increasing signal length. The theoretical curve (from Section 4.3) matches the empirical curve very well.

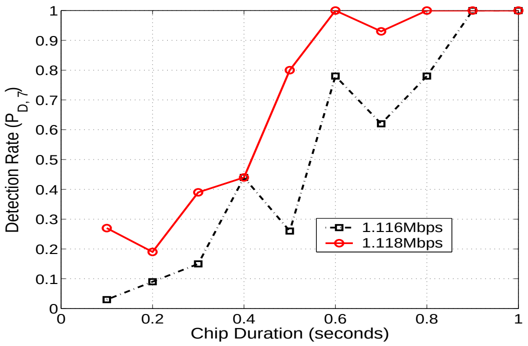

Figure 10. Detection Rate v.s. Chip Duration

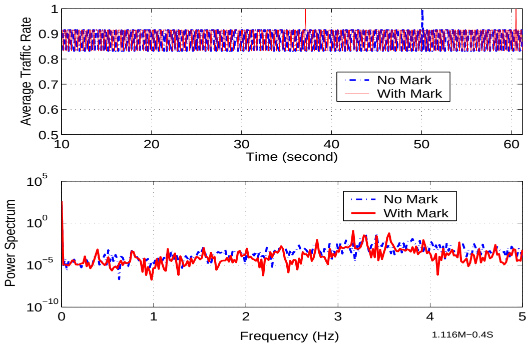

Figure 11. Overlapping Traffic Rate Curves

5.6 Flow Marking against Mixes with Different Mix Strategies

Figure 14 shows the detection rate in terms of CBR interference traffic rate for mixes with different batching strategies. We use node n_5 and n_7 as mix nodes in Figure 7. The average delay of packets for timed mixes and continuous-time mixes are set at 5 ms.

We make the following observations from Figure 14. In the presence of batching strategies in the mix network, the DSSS-based flow marking can still achieve a detection rate of 100%. This demonstrates the effectiveness of the DSSS-based flow marking against anonymous communication systems with sophisticated strategies. Different mix strategies have different impact on the detection rate. A large interference traffic rate is needed to achieve a good detection rate for the continuous-time mix. This is because the continuous-time (stop-and-go) mix's reordering strategy is very aggressive. It reduces TCP throughput significantly. So we have to increase the interference traffic rate in order to interfere with the FTP traffic to preserve delectability.

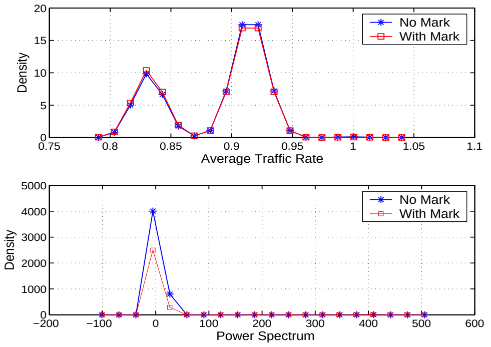

Figure 12. PDF of Overlapping Traffic Rate Curves

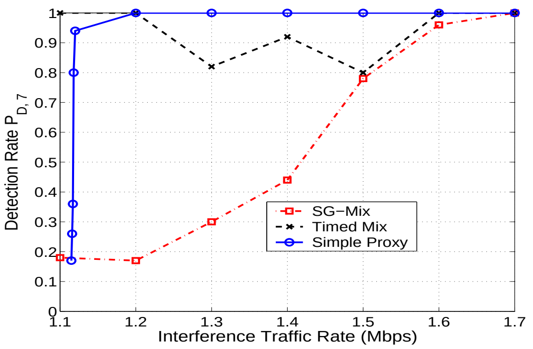

Figure 14. Detection Rate with Different Batching Strategies

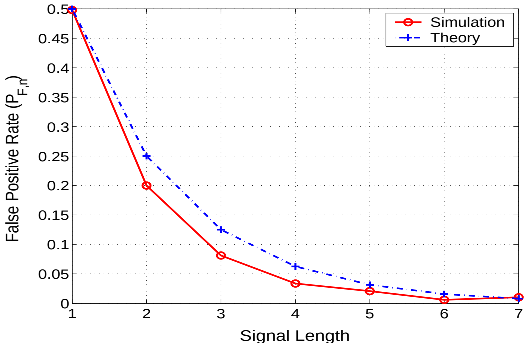

Figure 13. False Positive Rate

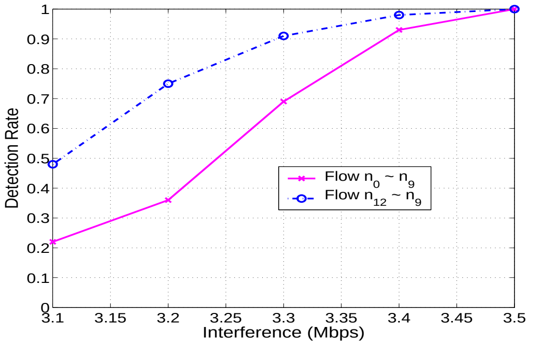

Figure 15. Detection Rate v.s. Interference Traffic Rate for Tracing Multiple Flows

5.7 Effectiveness of Tracing Multiple Flows

To demonstrate the effectiveness of DSSS-based flow marking technique at tracing multiple flows, we add another target flow of interest in Figure 7. The two target flows are: one from n_0 to n_9 and the other from n_{12} to n_9. Both flows traverse mixes n_5 and n_7. To interfere with these two flows, we use two CBR traffic sources, which are modulated using two different PN codes of length 7. One is from n_1 to n_4 and other one is from n_{13} to n_{10}.

Figure 15 shows the detection rate for both flows with respect to CBR interference traffic rates. In this set of simulations, we change all the 2Mbps links to 4Mbps links. The change allows TCP flows to get decent rate. The chip duration is 0.4 seconds and the signal of length 7. From this figure, we observe that the DSSS-based flow marking technique can effectively correlate senders and receivers of both flows. The reason is that the marked traffic flows are modulated by PN codes that are minimally cross-correlated (described in Section 3.3.2 and Appendix B of [24]). The capability of simultaneously tracing multiple flows can significantly improve the traceback capacity and investigators efficiency.

6 Evaluation by Tor

6.1 Experiment Setup

To evaluate the performance of the DSSS-based flow marking technique, we conducted extensive real-world experiments on *Tor*, a popular anonymous communication system. Figure 16 gives the experimental setup similar to that in Figure 7. We setup a server on Redhat 7.3 at one university campus location, and download a file from this server to an off-campus computer running with Fedora Core 3. This represents a typical use of *Tor* for anonymous file downloading and web browsing. The downloading software was the command line utility *wget*, shipped with most Linux distributions. By configuring wget's parameters of *http_proxy* and *ftp_proxy*, we can force *wget* to download files through *Privoxy*, the proxy server used by *Tor*.

For testing purposes, we used a computer to send out an appropriate volume of traffic which acts as interference to the server, and another computer was used as a *sniffer* to collect the traffic destined for the client machine. The *interferer* and server were connected by a hub, as were the *sniffer* and the client machine. We emphasize that there are more efficient approaches for interference (such as dropping packets) and sniffing available to authorized law enforcement agencies conducting Internet surveillance. We use this simple approach to demonstrate the effectiveness of the DSSS-based flow marking technique, even though the interference is not efficient.

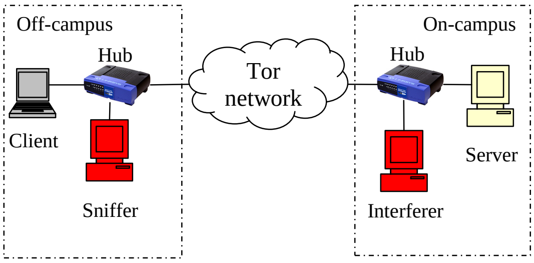

Figure 16. Experiment Setup

An accurate timer is critical for DSSS-based flow marking. The accurate timer is needed when we use the spread signal to modulate a TCP flow and embed the spread signal accurately into the TCP flow at the transmitter. A coarse timer will inhibit synchronization of the PN code at the receiver. In our experiments, we used the high resolution timer patch [26] to Linux kernel 2.6.17 and used POSIX 1003.1b Section 14 (Clocks and Timers) API for the *interferer* to produce accurate timers. The *interferer's* machine was a Gateway E6500 with a 2.8GHz CPU. The timer accuracy we achieved was roughly 0.1ms, which is reasonable in our context. Our future work includes testing the DSSS-based flow marking technique using Real Time Linux systems with their more accurate timing [27].

6.2 Experiment Results

We developed a prototype traceback software using Matlab. Figure 17 shows the interface of this software. The interface is much like that of a seismograph. A traffic trace is fed into the traceback software, which uses the matched filter based approach (described in Section 3.3.1) to recover the predefined signal, in our case, {*1 -1 1 1 -1 1 -1*}. If there is a match with this signal, we print out the *time* of the match above the corresponding time axis position and at an appropriate height to distinguish different matches[4]. In this figure, we show one matched signal near time 2400 seconds. In the experiments for Figure 17, chip duration is *2* seconds[5], and the PN code is an m-sequence code of length 7, {*1 -1 -1 1 1 1 -1*}. For every 196 seconds, we replay the flow marking against the file download traffic. Based on our experiments on *Tor*, we make the following observations.

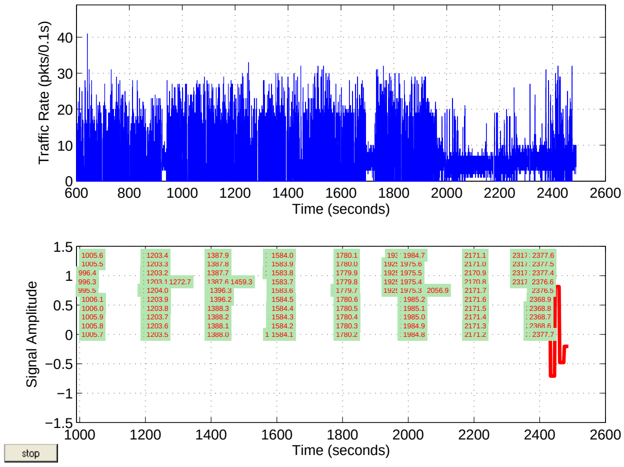

Figure 17. Demo of Tracking Anonymous Flows through Tor Mix Network

1. The interval between two recovered continuous signals is indeed approximately 196 seconds (the gap between columns of matched points). This demonstrates the effectiveness of the DSSS-based flow marking technique against *Tor*. It can be observed that there are columns of matches forming bars. This is because the sampling interval is 0.1 seconds and we attempt to match the target traffic every 0.1 seconds. Since the chip duration of 2 seconds is much larger than the sample interval of 0.1 seconds, the signal can be recovered even if the synchronization is somewhat skewed. Intuitively, a high bar indicates a strong signal match. The DSSS-based flow marking technique is invisible to people without knowledge of the PN code and embedded signal. Here we only show the the curve of the time domain traffic rate and similar results in Section 5.4 are also obtained (e.g., no any useful pattern showing that an ongoing traceback process exists).

2. The traffic rate in *Tor* may vary dramatically. *Tor* is a heterogeneous network built with a variety of *Tor* servers, composed of whatever users contribute. The performance and bandwidth of those servers vary dramatically. Also, *Tor* creates a mix routing path dynamically: a particular flow will continue to use the same routing path, otherwise the path is changed every 10 minutes for a *Tor* client. This may lead to wide performance differences from time to time. Figure 18 shows the Cumulative Distribution Function (CDF) of the average file download rate and demonstrates the performance variation. We download a file of size 455,902 bytes and record the downloading time for each run. The average rate is calculated as the file size over

[4] Each time stamp represents a different match. The vertical position of the time stamp is not significant and, in particular, is not related to the signal amplitude.

[5] This is reasonable given the low traffic rate in *Tor*. Refer to Figure 18.

the download time. We conducted the experiment for 10 days from July 12, 2006 to July 21, 2006, and collected the download time every 10 minutes. As shown in Figure 18, we observe that 70% of the flows had a traffic rate lower than 100KB/s and 20% had a traffic rate lower than 20KB/s.

3. Although there are wild rate dynamics for traffic using *Tor*, the DSSS-based flow marking technique is still effective. In Figure 17, we can see that even the flow rate along the same path has wild dynamics. However, our DSSS-based flow marking technique can still effectively recover the signal.

4. The DSSS-based flow marking technique sometimes misidentified the user input signal. We can see bars appearing away from the expected times. For example, at time 1272 seconds, we mistakenly matched the input signal. However, most such false positives are easily recognized and discarded since they are inconsistent with the expected propagation delays. The *interferer* and *sniffer* can synchronize with each other using the Network Time Protocol (NTP) or Global Positioning System (GPS). Then the *sniffer*, who is operating the traceback software, knows roughly the time when the signal should arrive. The propagation delay can be derived statistically through experiments with *Tor*. Thus, a positive signal match beyond that propagation delay is false and can be discarded. This is the approach we use to calculate the detection rate in Figure 19.

Figure 19 shows the detection rate of a 7-bit signal modulated by a PN code of length 7 in terms of chip duration. When we obtained the detection rate in Figure 19, we didn't apply any offline training for the decision rule. From Figure 19, we can make the following observation consistent with those from Figure 10: a long chip duration can increase the mark amplitude A in (12), and achieve a greater SNR. The greater SNR leads to a better detection rate. In reality, a longer chip duration is not always good. There exists a variety of unexpected interference on the Internet. A longer chip duration increases the possibility that impulse-like noise (bursts of noise interference) may occur during a chip. This will destroy the signal and reduce the detection rate.

7 Related Work

In this section, we first introduce various work related to mix systems. Then we explore work related to degrading the anonymous communication through mix systems.

Mix techniques can be used for either message-based (high-latency) or flow-based (low-latency) anonymity applications. The message-based anonymity application can use relay servers, i.e., *mixes*, to reroute messages, which are encrypted by mixes' public keys. Mixes use source routing for message forwarding. Examples of message-based anonymity systems include the first Internet anonymity system *remailer* by Helsingius [11], *cypherpunk remailer* by Eric Hughes and Hal Finney [28], *Babel* by Gülcü and Tsudik [29] and *Mixmaster* by Cottrell [30]. Danezis et al. [2] recently developed a so-called Type III Anonymous Remailer Protocol *Mixminion*, which considers a relatively complete set of threats that researchers have discovered.

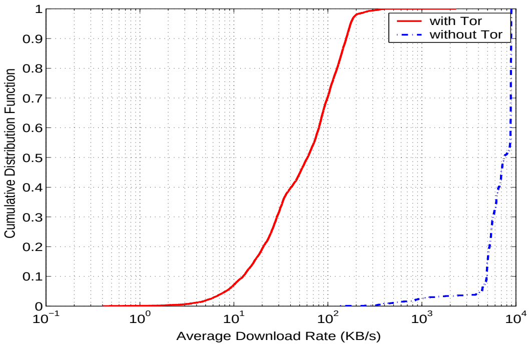

Figure 18. Average Download Rate in Tor Mix Network

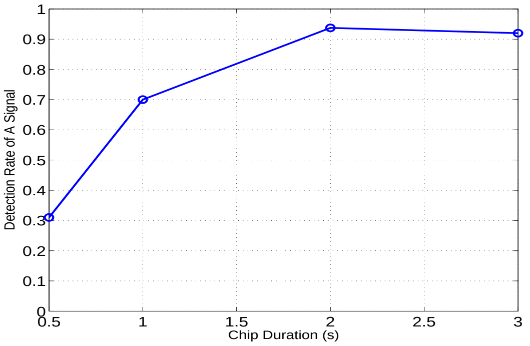

Figure 19. Detection Rate of a 7-bit Signal

Low-latency anonymity applications can use either core mix networks or peer-to-peer networks. In a system using a core mix network, users connect to a pool of mixes and select a forwarding path through the mix network to the receiver. *Tor* [4], *Freedom* [31], and many others belong to this category. In a system using a peer-to-peer network, every node is a mix, but it can also be a sender and/or receiver. A peer-to-peer mix network may scale well and provide better anonymity if enough participants contribute to and choose the anonymity service. *Crowds* [3], *Tarzan* [32], and many others belong to this category.

There have been many articles on how to degrade the anonymous communication through mix networks. To identify the communication relationship (e.g., find whether Alice is communicating with Bob) through a mix network, the similarity between Alice's outbound traffic and Bob's inbound traffic may be measured. Zhu et al. [8] proposed using mutual information for the similarity measurement.

In the one-mix case, an adversary collects a sample from an input flow and each output flow of the mix. Each sample is divided into multiple equally sized segments based on time. The number of packets in each segment is counted and forms a time series of packet counts. Then, the adversary chooses the output link, whose flow's packet count time series has the biggest mutual information with the input flow's packet count time series as the input flow's output link. To counter this scheme, the defensive strategy of using adaptive padding was discussed in [8]. Levine et al. [9] also tackled the same problem of identifying the communication relationship through a mix network for flow-based applications. However, they adopted the scheme that uses cross correlation to measure similarity between flows. If the cross correlation is beyond a threshold, the adversary decides the communication relationship; otherwise not. The choice of threshold is the key problem of this scheme and may not be easily derived in practice. To counteract this scheme, Levine et al. also proposed the defensive dropping strategy, e.g., Alice generating dummy packets to Bob and intermediate mixes on the flow's path randomly dropping those dummy packets.

Authors in [33] and [34] mentioned very briefly that a "spike" may be introduced into traffic to find the communication relationship between users, but did not pursue an in-depth study of how to introduce spikes, what kind of spike should be introduced, or how to recognize the spike. Zhang and Paxson proposed a simple ON/OFF based approach to correlated encrypted traffic [35]. Fu et al. [6] studied the flow marking scheme using the frequency as the unique pattern to degrade the anonymous systems and also pointed out that this marking scheme can be effectively countered by filter-based countermeasures (i.e., filtering out the suspect band of feature frequencies). Overlier et al. [36] studied the scheme to use one compromised mix node to identify the "hidden server" anonymized by Tor. However, their scheme relies on compromising at least one mix node of mix network. Also their scheme can be effectively countered by using random entry guard mix nodes and other means. Murdoch et al. [34] investigated timing based attack on Tor with similar assumptions of using compromised Tor nodes. Wang and Peng et al. [37, 7] studied the inter-packet timing based scheme to trace connections through stepping stones or anonymous VoIP systems. By slightly changing the timing of packets, that approach correlates communication relationship for encrypted network connections. However, that approach would not effectively track communication through mix network with batching strategies that manipulate inter-packet delivery timing.

In this paper, we studied a novel class of DSSS-base flow marking technique to effectively track anonymous communication through mix networks. Our proposed technique is capable of invisible traceback and does not require any knowledge of mix networks. Work in [38] also discussed the way to use PN code to secretly locate Internet threat monitors. Since it is applied to a different problem domain, the solution in [38] is much different from that in this paper, including the way of using the PN code, designed algorithms, decision rule, and theoretical analysis.

8 Conclusion

In this paper, we investigated how suspect malicious anonymous communication (ranging from audio and video plagiarists to potential terrorists) can be effectively identified and traced on the Internet. We developed a DSSS-based network flow marking technique for traceback, introducing invisible marks into a target traffic flow which is persist and detectable despite a variety of anonymity schemes, making it possible to trace anonymous communications. Through a combination of analytical modeling and an extensive set of simulations, we demonstrated the effectiveness of the DSSS-based flow marking technique. A proof-of-concept traceback system was also developed, and the analytical and simulation results were verified by empirical data from Internet experiments using *Tor*.

The DSSS-based flow marking technique has the following advantages. It can achieve both high detection and low false positive rates. The high detection rate originates from the flexible selection of parameters such as the PN code length, signal length, mark amplitude, and chip duration. Furthermore, the designed DSSS-based flow marking system has a simple decision rule and does not require the extensive training required by many other approaches.

We emphasize that our DSSS-based technique is a general one and can be used to trace other communications, under any conditions where the IP header information is unreliable for recognizing communication relationships. To the best of our knowledge, this paper is the first to apply spread spectrum techniques for the purpose of Internet traceback. As part of our future work, we are applying DSSS techniques to other realms and evaluate their effectiveness.

References

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

[2] 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.

[3] M. Reiter and A. Rubin, "Crowds: Anonymity for web transactions," *ACM Transactions on Information and System Security*, vol. 1, no. 1, 1998.

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

[5] R. Dingledine and N. Mathewson, “Tor: An anonymous internet communication system,” http://tor.eff.org/index.html.en, 2006.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] A. Serjantov, R. Dingledine, and P. Syverson, “From a trickle to a flood: active attacks on several mix types,” in *Proceedings of Information Hiding Workshop (IH)*, February 2002.

[11] J. Helsingius, “Press release: Johan helsingius closes his internet remailer,” http://www.eff.org/Censorship/Foreign_and_local/Finland/960830_penet_closure.announce, August 1996.

[12] IEEE Computer Society, *Part 11: Wireless LAN Media Access Control (MAC) and Physical Control Specifications (802.11)*, IEEE, 1999.

[13] R. K. Pickholtz, D. L. Schilling, and L. B. Milstein, “Theory of spead-spectrum communication - tutorial,” *IEEE Transaction on Communication*, vol. 30, no. 5, pp. 855–884, 1982.

[14] E. J. Crusellers, M. Soriano, and J. L. Melus, “Spreading codes generator for wireless cdma network,” *International Journal of Wireless Personal Communications*, vol. 7, no. 1, 1998.

[15] M. Bellare, S. Goldwasser, and D. Miccianciom, “Pseudo-random number generation within cryptographic algorithms: the dss case,” in *Proceedings of advances in cryptology’97, Lecture Notes in Computer Science*, May 1997.

[16] L. Wang and B. B. Hirsbrunner, “Pn-based security design for data storage,” in *Proceedings of Databases and Applications*, Feberary 2004.

[17] X. G. Xia, C. G. Boncele, and G. R. Arce, “A multiresolution watermark for digital images,” in *Proceedings of International Conference on Image Processing (ICIP)*, October 1997.

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

[19] M. Fomenkov, K. Keys, D. Moore, and K. Claffy, “Longitudinal study of internet traffic in 1998-2003,” in *Proceedings of the Winter International Synposium on Information and Communication Technologies*, January 2004.

[20] K. Thompson, G. Miller, and R. Wilder, “Wide-area internet traffic patterns and characteristics,” *IEEE Network magazine*, vol. 11, no. 6, November/December 1997.

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

[22] G. Buracas, “m-sequence generation program,” http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=990&objectType=file, 2003.

[23] A. Kuzmanovic and E. W. Knightly, “Low-rate tcp-targeted denial of service attacks,” in *Proceedings of ACM SIGCOMM*, August 2003.

[24] W. Yu, X. Fu, S. Graham, D. Xuan, and W. Zhao, “DSSS-based flow marking technique for invisible traceback,” Technical report TR-0307-001, College of Business and Information Systems, Dakota State University, March 2007.

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

[26] G. Anzinger, “High resolution timers,” http://www.tglx.de/projects/hrtimers/2.6.17/, 2006.

[27] timesys.com, “Embedded linux development tools,” http://www.timesys.com/, 2006.

[28] S. Parekh, *Prospects for Remailers - Where is Anonymity Heading on the Internet*, http://www.firstmonday.dk/issues/issue2/remailers/, 1996.

[29] C. Gülcü and G. Tsudik, “Mixing E-mail with Babel,” in *Proceedings of the Network and Distributed Security Symposium (NDSS)*, February 1996.

[30] U. Möller and L. Cottrell, *Mixmaster Protocol — Version 2*, http://www.eskimo.com/ rowdenw/crypt/Mix/draft-moeller-mixmaster2-protocol-00.txt, January 2000.

[31] P. Boucher, A. Shostack, and I. Goldberg, “Freedom systems 2.0 architecture,” December 2000.

[32] M. J. Freedman and R. Morris, “Tarzan: A peer-to-peer anonymizing network layer,” in *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS)*, November 2002.

[33] A. Serjantov and P. Sewell, “Passive attack analysis for connection-based anonymity systems,” in *Proceedings of European Symposium on Research in Computer Security (ESORICS)*, October 2003.

[34] 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.

[35] Y. Zhang and V. Pasxon, “Detecting stepping stones,” in *Proceedings of the 9th USENIX Security Symposium*, August 2000.

[36] L. Overlier, “Locating hidder servers,” in *Proceedings of the IEEE Security and Privacy Symposium (S&P)*, May 2006.

[37] 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.

[38] W. Yu, X. Wang, X. Fu, D. Xuan, and W. Zhao, “Towards secret fingerprint of internet threat monitors,” Technical report, Computer and Engineering Dept., The Ohio State University, Septermber 2006.

[39] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing*, PrenticeHall, Inc., 1975.

[40] R. Dixon, *Spread Spectrum Systems, 2nd Edition*, John Wiley & Sons, 1984.

Appendix A: Proof of Theorem 1

Proof 1 *Recall T_s is the symbol duration and T_c is the chip duration. We impose the condition that*

$$T_s = N_c T_c, \tag{18}$$

where N_c is the length of PN code. We consider the case that there does not exist any noise. Therefore, as we discussed in Section 3.1,

$$rx = Ad_t \cdot c_t + D, \tag{19}$$

where d_t is the signal (either $+1$ or -1), A is the mark amplitude, and D is the direct component. Let's use tx_b to represent $Ad_t \cdot c_t$ in (19), so

$$rx = tx_b + D, \tag{20}$$

where tx_b is the base band signal the receiver intends to recover.

The sheer purpose of the high-pass filter at the receiver is to remove the direct current component of the received signal rx. From Discrete Fourier Transform (DFT) [39], we have

$$RX(k) = \sum_{n=0}^{N_c-1} (tx_b(n) + D) W_N^{kn} \tag{21}$$

$$= TX_b(k) + \sum_{n=0}^{N_c-1} D W_N^{kn}, \tag{22}$$

$$rx(n) = \frac{1}{N_c} \sum_{k=0}^{N_c-1} RX(k) W_N^{-kn}, \tag{23}$$

where

$$W_N = e^{-j2\pi/N_c}. \tag{24}$$

If the direct current component $RX(0)$ is removed by the low-pass filter, then

$$rx'(n) = \frac{1}{N_c}[\sum_{k=0}^{N_c-1} RX(k) W_N^{-kn} - RX(0)] \tag{25}$$

$$= \frac{1}{N_c}[\sum_{k=0}^{N_c-1} (TX_b(k) + \sum_{m=0}^{N_c-1} D W_N^{km}) W_N^{-kn} - RX(0)] \tag{26}$$

$$= \frac{1}{N_c}[\sum_{k=0}^{N_c-1} (TX_b(k) W_N^{-kn} + N_c D - RX(0)]. \tag{27}$$

If c_t is from a m-sequence generator, the length of c_t, N_c, is always an odd number. Thus,

$$RX(0) = \sum_{n=0}^{N_c-1} (tx_b(n) + D) W_N^{0n} \tag{28}$$

$$= \sum_{n=0}^{N_c-1} Ad_t c_t(n) + N_c D \tag{29}$$

$$= Ad_t \sum_{n=0}^{N_c-1} c_t(n) + N_c D. \tag{30}$$

According to the "balanced" property of PN code [40], i.e., the occurrence frequencies of -1 and $+1$ are nearly equal. Without loss of generality, we assume that $\sum c_r = 1$. Then we have

$$RX(0) = Ad_t + N_c D, \tag{31}$$

and

$$rx'(n) = \frac{1}{N_c}[\sum_{k=0}^{N_c-1} TX_b(k) W_N^{-kn} - Ad_t] \tag{32}$$

$$= tx_b(n) - \frac{Ad_t}{N_c}. \tag{33}$$

Therefore,

$$\sum rx_b = \sum (rx' \cdot c_r) \tag{34}$$

$$= \sum (tx - \frac{Ad_t}{N_c}) \cdot c_r \tag{35}$$

$$= \sum (Ad_t c_t - \frac{Ad_t}{N_c}) \cdot c_r \tag{36}$$

$$= AN_c d_t - \frac{Ad_t}{N_c} \sum c_r \tag{37}$$

$$= Ad_t (N_c - \frac{1}{N_c}). \tag{38}$$

Denote rx_{+1} as the result of $\sum rx_b/N_c$ when the 1-bit signal d_t is $+1$ and rx_{-1} as the result of $\sum rx_b/N$ when the 1-bit signal d_t is -1. Therefore, after despreading, we have

$$rx_{+1} = \sum r_b|_{d_t=1}/N_c \tag{39}$$

$$= A(N_c - \frac{1}{N})/N_c \tag{40}$$

$$= A(1 - 1/N_c^2), \tag{41}$$

$$rx_{-1} = \sum rx_b|_{d_t=-1}/N_c \tag{42}$$

$$= -A(N_c - \frac{1}{N_c})/N_c \tag{43}$$

$$= -A(1 - 1/N_c^2) \tag{44}$$

$$= -rx_{+1}. \tag{45}$$

Appendix B: Effectiveness of Tracing Multiple Flows

We now formally analyze the effectiveness of the DSSS-based flow marking on tracing multiple flows through a mix. Refer to Figure 6, we use one PN code, $c_{t,1}$, to modulate flow 1 on input link 1 and a second PN code, $c_{t,2}$, to modulate the flow 2 on input link 2. Without loss of generality, we assume that *sniffer* tries to determine which flows go through output link 2.

Let's use the $d_{t,1}$ to represent the signal for flow 1 and $d_{t,2}$ to represent the signal for flow 2, respectively. Let's use tx_1 to represent an embedded spread signal for flow 1 modulated by $c_{t,1}$ and tx_2 to represent an embedded spread signal for flow 2 modulated by $c_{t,2}$. From the analysis in Section 3, we have

$$\begin{aligned} tx_1 &= A_1 d_{t,1} c_{t,1}, && (46) \\ tx_2 &= A_2 d_{t,2} c_{t,2}. && (47) \end{aligned}$$

As we can see that tx_1 and tx_2 are mixed on output link 2. Assume that PN codes for these two signals are synchronized (using the scheme discussed in Section 3.3.1). After the received integrated signals on output link 2 pass through the high-pass filter, we have

$$\begin{aligned} rx'_1(n) &= tx_1(n) - A_1 c_{t,1}/N_c, && (48) \\ rx'_2(n) &= tx_2(n) - A_2 c_{t,2}/N_c. && (49) \end{aligned}$$

Substitute (46) and (47) into (48) and (49) respectively, we have

$$\begin{aligned} rx'_1 &= A_1 d_{t,1} c_{t,1} - A_1 d_{t,1}/N_c, && (50) \\ rx'_2 &= A_2 d_{t,2} c_{t,2} - A_2 d_{t,2}/N_c. && (51) \end{aligned}$$

To determine if flow 1 on input link 1 passes through output link 2, we use $c_{r,1} = c_{t,1}$ to despread the integrated flow on output link 2,

$$\begin{aligned} rx_{+1} &= rx'_1\, c_{r,1} + rx'_2 c_{r,1} && (52) \\ &= A_1(1-1/N_c^2) + A_2 d_{t,2} c_{t,2} \cdot c_{r,1}/N_c - A_2 d_{t,2}/N_c^2 && (53) \\ rx_{-1} &= -A_1(1-1/N_c^2) + A_2 d_{t,2} c_{t,2} \cdot c_{r,1}/N_c - A_2 d_{t,2}/N_c^2 && (54) \end{aligned}$$

Since $c_{t,1}$ and $c_{t,2}$ are minimally cross-correlated, e.g., $c_{t,2} \cdot c_{r,1} \approx 0$, we have

$$\begin{aligned} rx_{+1} &= A_1(1-1/N_c^2) - A_2 d_{t,2}/N_c^2, && (55) \\ rx_{-1} &= -A_1(1-1/N_c^2) - A_2 d_{t,2}/N_c^2. && (56) \end{aligned}$$

From (55) and (56), the original signal $d_{t,1}$ can be accurately recovered by using PN code $c_{t,1}$. Similarly, the original signal $d_{t,2}$ can also be accurately recovered by using PN code $c_{t,2}$.

As we can see, the low cross-correlation property of PN code provides salient capability for investigators to trace multiple flows through the same mix without interfering with each other. Note that PN code by the m-sequence generator used in the paper has good cross-correlation property. Investigating performance of other codes such as gold codes [21] will be part of our on-going work.

Appendix C: Proof of Lemma 1

Proof 2 *First, when White Gaussian Noise (WGN) multiplies the receiver's local spreading code c_r, it will be still WGN. Since our classifier is $\sum r_b/N_c$, its noise component is denoted as $w_r = \sum w \cdot c_r/N_c$. Let's calculate w_r's mean and variance as follows:*

$$\begin{aligned} E(w_r) &= E(\textstyle\sum w \cdot c_r/N_c) && (57) \\ &= \textstyle\sum E(w_i)E(c_r)/N_c. && (58) \end{aligned}$$

As we know, $E(w_i) = 0$, then

$$E(w_r) = 0. \quad (59)$$

and

$$\begin{aligned} var(w_r) &= E((w_r - E(w_r))^2) && (60) \\ &= E((\textstyle\sum w \cdot c_r/N_c)^2) && (61) \\ &= \textstyle\sum (E(n_i^2)E(c_{r,i}^2))/N_c^2. && (62) \end{aligned}$$

Since $E(w_i^2) = \sigma_w^2$ and $E(c_r^2) = 1$ (given that a chip of the PN code takes 1 or -1 with equal probability), we have

$$var(w_r) = \frac{\sigma_w^2}{N_c}. \quad (63)$$

For different symbols 1 and -1 of the signal, we have

$$\begin{aligned} rx_{+1} &= A(1-1/N_c^2) + w_r, && (64) \\ rx_{-1} &= -A(1-1/N_c^2) + w_r. && (65) \end{aligned}$$

To differentiate rx_{+1} and rx_{-1}, we can use Bayes decision rule as shown in Figure 20. The decision boundary of 0, shown in (10), achieves the minimum decision error.

Assume symbols $+1$ and -1 have the same occurrence probability. The decision error P_e, the probability that a signal is not recognized, can be easily derived as the area of the shaded region in Figure 20.

$$\begin{aligned} P_e &= \frac{1}{2}P(rx_{+1} < 0) + \frac{1}{2}P(rx_{-1} > 0) && (66) \\ &= \int_0^\infty \frac{1}{\sqrt{2\pi}\sigma_w/\sqrt{N_c}} e^{\frac{[x+A(1-1/N_c^2)]^2}{2\sigma_w^2/N_c}} dx && (67) \\ &= \int_{\frac{A(1-1/N_c^2)}{\sigma_w/\sqrt{N_c}}}^\infty \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx && (68) \\ &= \frac{1}{2} erfc(\sqrt{\varepsilon}), && (69) \end{aligned}$$

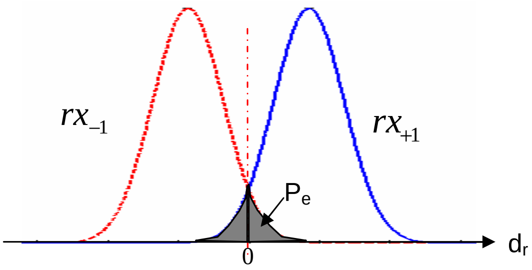

Figure 20. Bayes Decision Rule

where $erfc(.)$ is the complementary error function and monotonically decreases with ε, and

$$\varepsilon = \frac{(N_c^2-1)^2}{2N_c^3}\frac{A^2}{\sigma_w^2}. \tag{70}$$

Since the detection rate P_D for 1-bit original signal is as the probability that one symbol is correctly recognized, we have

$$\begin{aligned} P_D &= 1-P_e & (71)\\ &= 1-\frac{1}{2}erfc(\sqrt{\varepsilon}). & (72)\end{aligned}$$

Appendix D: Proof of Corollary 1

Proof 3 *When $SNR = A^2/\sigma_w^2 = 0$, we have $\varepsilon = 0$ in (12). Therefore,*

$$\begin{aligned} erfc(\sqrt{\varepsilon}) &= erfc(0) = 1, & (73)\\ P_e &= \frac{1}{2}erfc(\sqrt{\varepsilon}) = 0.5, & (74)\\ P_D &= 1-P_e = 0.5. & (75)\end{aligned}$$