RAINBOW: A Robust And Invisible Non-Blind Watermark for Network Flows

Amir Houmansadr* Negar Kiyavash† Nikita Borisov*
*Dept. of Electrical and Computer Engineering
†Dept. of Industrial and Enterprise Systems Engineering
University of Illinois at Urbana–Champaign
{ahouman2,kiyavash,nikita}@illinois.edu

Abstract

Linking network flows is an important problem in intrusion detection as well as anonymity. Passive traffic analysis can link flows but requires long periods of observation to reduce errors. Watermarking techniques allow for better precision and blind detection, but they do so by introducing significant delays to the traffic flow, enabling attacks that detect and remove the mark, while at the same time slowing down legitimate traffic. We propose a new, non-blind watermarking scheme called RAINBOW that is able to use delays hundreds of times smaller than existing watermarks by eliminating the interference caused by the flow in the blind case. As a result, our watermark is invisible to detection, as confirmed by experiments using information-theoretic detection tools.

We analyze the error rates of our scheme based on a mathematical model of network traffic and jitter. We also validate the analysis using an implementation running on PlanetLab. We find that our scheme generates orders of magnitudes lower rates of false errors than passive traffic analysis, while using only a few hundred observed packets. We also extend our scheme so that it is robust to packet drops and repacketization and show that flows can still be reliably linked, though at the cost of somewhat longer observation periods.

1 Introduction

Internet attackers commonly relay their traffic through a number of (usually compromised) hosts in order to hide their identity. Detecting such hosts, called stepping stones, is therefore an important problem in computer security. The detection proceeds by finding correlated flows entering and leaving the network. Traditional approaches have used patterns inherent in traffic flows, such as packet timings, sizes, and counts, to link an incoming flow to an outgoing one [17, 24, 8, 20, 3]. More recently, an active approach called *watermarking* has been considered [21, 16]. In this approach, traffic characteristics of an incoming flow are actively perturbed as they traverse some router to create a distinct pattern, which can later be recognized in outgoing flows. These techniques also have relevance to anonymous communication, as linking two flows can be used to break anonymity, and both passive traffic analysis [12, 6] and active watermarking [18, 19, 23] have been studied in that domain as well.

The choice between passive and active techniques for traffic analysis exhibits a tradeoff. Passive approaches require observing relatively long-lived network flows, and storing or transmitting large amounts of traffic characteristics. Watermarking approaches are more efficient, with shorter observation periods necessary. They are also *blind*: rather than storing or communicating traffic patterns, all the necessary information is embedded in the flow itself. This, however, comes at a cost: to ensure robustness, the watermarks introduce large delays (hundreds of milliseconds) to the flows, interfering with the activity of benign users, and making them subject to attacks [13, 11].

Motivated by this, we develop a new scheme for linking flows, called RAINBOW. As with passive techniques, our scheme will record traffic timings of incoming flows and correlate them with outgoing flows. However, we also insert a watermark value by delaying some packets. As the watermark is generated independently of the flows, this will diminish the effect of natural similarities between two unrelated flows, and allow a flow linking decision to be made over a much shorter time period. We use spread-spectrum techniques to make our delays much smaller than previous work. We use delays that are on the order of only a few milliseconds; this means that our watermarks not only do not interfere with traffic patterns of normal users, they are also virtually *invisible*, since the delays are of the same magnitude as natural network jitter.

We analyze our technique using a mathematical model of network traffic and delays. We show that in our tech-

nique, low-amplitude watermarks can achieve false-positive and false-negative rates that are an order of magnitude smaller than passive traffic analysis with short observation periods—a few hundred packets. We validate our analysis by building a prototype implementation of our scheme. We test it by generating flows with timings taken from real SSH [22] traffic traces, and linking flows that traversed the Internet between PlanetLab [1] nodes. Our scheme performed quite well in this setting as well. Note that PlanetLab introduces significantly more jitter than would be present in an enterprise network, so in practice, much lower watermark delays, or smaller packet sizes, can be used. We also analyze the invisibility of our scheme by subjecting it to several information-theoretic detection tools [9, 13].

We also extend our scheme to handle dropped or inserted packets. Such changes to flows will occur naturally due to packet losses, retransmissions, or repacketization. By adjusting our scheme to perform *selective correlation*, where packets that do not match up between the incoming and outgoing flows are dropped, our scheme can be made robust to packets being inserted and deleted, though at the cost of either longer observation periods or higher watermark amplitude.

The rest of this paper is organized as follows: we review the problem of stepping stone detection and existing schemes in Section 2. The RAINBOW scheme is presented in Section 3. In Section 4, we use *detection theory* to analyze performance of the proposed scheme. We provide implementation results in Section 5, validating the analysis. In Section 6 we extend RAINBOW by introducing *selective correlation* to make it robust to flow modifications. Discussions on watermark invisibility are presented in Section 7, and paper is concluded in Section 8 along with some future research directions.

2 Background

In this section, we review the problem of detecting stepping stones and then review both the passive and active approaches to the problem. We compare the advantages and disadvantages of the two techniques, motivating our approach.

2.1 Stepping Stone Detection

A stepping stone is a host that is used to relay traffic through an enterprise network to another remote destination. Stepping stones are used to disguise the true origin of an attack. Detecting stepping stones can help trace attacks back to their true source. Also, stepping stones are often indicative of a compromised machine. Thus detecting stepping stones is a useful part of enterprise security monitoring.

Generally, stepping stones are detected by noticing that an outgoing flow from an enterprise matches an incoming flow. For example, in Figure 1(a), flow 2 will have the same characteristics as flow 5. Since the relayed connections are often encrypted (using SSH [22], for example), only characteristics such as packet sizes, counts, and timings are available for such detection. And even these are not perfectly replicated from an incoming flow to an outgoing flow, as they are changed by padding schemes, retransmissions, and jitter. As a result, statistical methods are used to detect correlations among the incoming and outgoing flows. We next review the passive and active approaches.

2.2 Passive Traffic Analysis

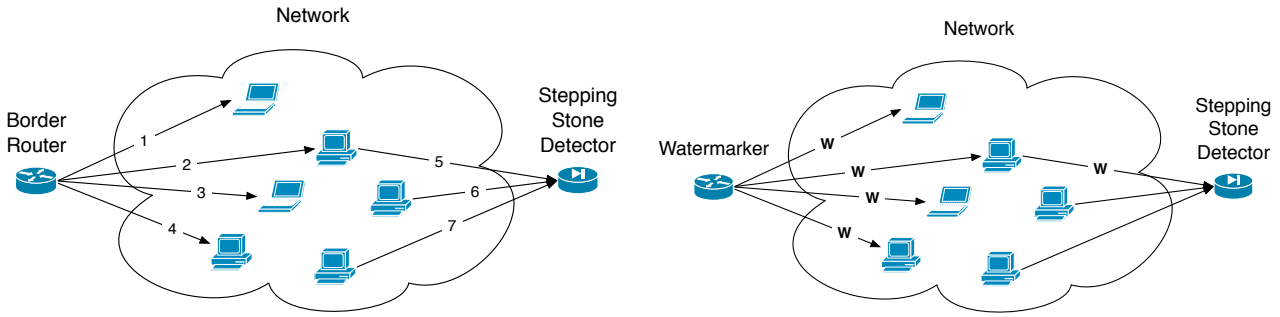
In general, passive traffic analysis techniques operate by recording characteristics of incoming streams and then correlating them with outgoing ones. The right place to do this is often at the border router of an enterprise, so the overhead of this technique is the space used to store the stream characteristics long enough to check against correlated relayed streams, and the CPU time needed to perform the correlations. In a complex enterprise with many interconnected networks, a connection relayed through a stepping stone may enter and leave the enterprise through different points; in such cases, there is additional communications overhead for transmitting traffic statistics between border routers.

The passive schemes have explored using various characteristics for correlating streams. Zhang and Paxson [24] model interactive flows as on–off processes and detect linked flows by matching up their on–off behavior. Wang et al. [20] focus on inter-packet delays, and consider several different metrics for correlation. More recently, He and Tong used packet counts for stepping stone detection [10].

Donoho et al. were the first to consider intruder evasion techniques [8]. They defined a *maximum-tolerable-delay* (MTD) model of attacker evasion and suggested wavelet methods to detect stepping stones while being robust to adversarial action. Blum et al. used a Poisson model of flows to create a technique with provable upper bounds on false positive rates [3], given the MTD model. However, for realistic settings, their techniques require thousands of packets to be observed to achieve reasonable rates of false errors.

2.3 Watermarks

To address some of the efficiency concerns of passive traffic analysis, Wang et al. proposed the use of watermarks [21]. In this scenario, a border router will modify the traffic timings of the incoming flows to contain a particular pattern—the watermark. If the same pattern is present in an outgoing flow, a stepping stone is detected. This can be seen in Figure 1(b).



(a) A stepping stone

(b) Network flow watermarks

Figure 1. Stepping Stone Detection

Watermarks improve upon passive traffic analysis in two ways. First, by inserting a pattern that is uncorrelated with any other flows, they can improve the detection efficiency, requiring smaller numbers of packets to be observed (hundreds instead of thousands) and providing lower false-positive rates (10^{-4} or lower, as compared to 10^{-2} with passive watermarks). Second, they can operate in a *blind* fashion: after an incoming flow is watermarked, there is no need to record or communicate the flow characteristics, since the presence of a watermark can be detected independently. The detection is also potentially faster, as here is no need to compare each outgoing flow to all the incoming flows within the same time frame.

Watermarking techniques for network flows have been based on existing techniques for multi-media watermarking. For example, Wang et al. based their scheme on QIM watermarks [4]. Two other watermark schemes [16, 19] are based on patchwork watermarking [2], and Yu et al. [23] developed one based on spread-spectrum techniques [5]. Some of the schemes target anonymous communication rather than stepping stones as the application area (both involve the problem of linking flows), but the techniques for both are comparable.

2.4 Watermark Properties

To motivate our design, we first propose some desirable properties of network flow watermarks. First of all, a watermark should be *robust* to modifications of the traffic characteristics that will occur inside an enterprise network, such as jitter. Watermarks should also be resilient to an adversary who actively tries to remove them from the flow, a property we call *active robustness*. The watermarks should also introduce little *distortion*, in that they should not significantly impact the performance of the flows. This is important because in a stepping-stone scenario, most watermarked flows will be benign. Finally, watermarks should be *invisible* even to attackers who specifically try to test for their presence.

Looking at previous designs, all of them fail to be invisible: the watermarks introduce large delays, on the order of hundreds of milliseconds, on some packets, which can be easily detected by an attacker [13]. In fact, they cannot even be considered low-distortion, as such large delays are easily noticeable and bothersome to legitimate users. The watermarks are also not actively robust, as demonstrated by recent attacks [13, 11].

We also observe that active robustness and invisibility are likely to be impossible to achieve at the same time. This is because to be invisible, the watermark can only introduce minute changes to the packet stream. In particular, it cannot introduce jitter of more than a few milliseconds, since otherwise it will be possible to tell it apart from the natural network jitter. However, an active attacker will be willing to introduce large delays to the network; for example, the maximum tolerable delay suggested in previous work is 500ms. As such, he will be able to destroy any low-order effects that will be introduced by the watermark.

Further, it is easy to imagine an attacker determined to hide his tracks using even more drastic measures, such as using dummy packets to generate a completely independent Poisson process [3], which will render any linking techniques ineffective. As such, we decided to design a watermark scheme that is robust to normal network interference, though not actively robust, and is invisible. This will serve to detect stepping stones where attackers are unwilling (or unable) to actively distort their stream as it crosses a stepping stone. Further, as the watermark will be invisible, attackers will not be able to tell if they are being traced and thus will be less likely to try to apply costly watermark countermeasures.

3 RAINBOW Watermark

We next present a design of a new watermark scheme we call RAINBOW, for Robust And Invisible Non-Blind

Watermark. Our scheme is robust (to passive interference) and invisible. However, to achieve invisibility while maintaining detection efficiency, we make the scheme *non-blind*; that is, incoming flows timings are recorded and compared with the timings of outgoing flows. This allows us to make a robust watermark test with even low-amplitude watermarks.

3.1 Watermark Embedding

In this section, we explain the watermark embedding process as shown in Figure 2. Suppose that a flow with the packet timing information $\{t_i^u | i = 1, .., n+1\}$ enters border router where it is to be watermarked (we use the superscript u to denote an "unwatermarked" flow). Before embedding the watermark, the inter-packet delays (IPDs) of the flow, $\tau_i^u = t_{i+1}^u - t_i^u$ are recorded in an IPD database, which is accessible by the watermark detector. The watermark is subsequently embedded by delaying the packets by an amount such that the IPD of the ith watermarked packet is $\tau_i^w = \tau_i^u + w_i$. The watermark components $\{w_i\}_{i=1}^n$ take values $\pm a$ with equal probability:

$$w_i = \begin{cases} +a & \text{w. p. } \frac{1}{2} \\ -a & \text{w. p. } \frac{1}{2} \end{cases} \quad (1)$$

The value a is chosen to be small enough so that the artificial jitter caused by watermark embedding is invisible to ordinary users and attackers[1].

In order to apply watermark delays on the flow, output packet t_i is delayed by $w_0 + \sum_{j=1}^{i-1} w_i$, where w_0 is the initial delay applied to the first packet. This results in $\tau_i^w = \tau_i^u + w_i$, as desired. Since we cannot delay a packet for a negative amount of time, w_0 must be chosen large enough to prevent this from happening. Since the sequence w_i is generated from a random seed, the watermarker can calculate all of the partial sums $\sum_{j=1}^{i-1} w_i$ in advance and adjust w_0 accordingly. If a particular random seed requires a very large initial delay w_0, a different seed can be chosen.

As the flow traverses the network, it accumulates extra delays. Let d_i be the delay that the packet accumulates by the time it reaches the watermark detector; i.e., the packet is received at the detector at time $t_i^r = t_i^w + d_i$. The IPD values at the detector are then:

$$\tau_i^r = t_{i+1}^r - t_i^r = \tau_i^u + w_i + \delta_i \quad (2)$$

where $\delta_i = d_{i+1} - d_i$ is the jitter present in the network.

3.2 Detection Scheme

Our detection scheme is non-blind and therefore the detector had access to the IPD database where the unwatermarked flows are recorded. Given an observed flow at the detector with IPDs $\tau^{\mathbf{r}}$ and a previously recorded flow $\tau^{\mathbf{u}}$, the detector must decide whether the two flows are linked or not. To do this, it computes the difference of the received and the recorded flow $\mathbf{y}$, where:

$$y_i = \tau_i^r - \tau_i^u \quad (3)$$

If the flows are in fact the same, then:

$$y_i = w_i + \delta_i \quad (4)$$

At this point, the watermark detection problem can be modeled as detecting a known spread spectrum signal $\mathbf{w}$ combined with noise given by network jitter, δ. Previous research in the multimedia area has used *normalized correlation* as an efficient detection scheme for spread-spectrum watermarks [5]. A normalized correlation is the inner product of two sequences, divided by their norms:

$$N(\mathbf{a}, \mathbf{b}) = \frac{\langle \mathbf{a}, \mathbf{b} \rangle}{||\mathbf{a}|| \cdot ||\mathbf{b}||} = \frac{\sum_{i=1}^n a_i b_i}{\sqrt{(\sum_{i=1}^n a_i^2)(\sum_{i=1}^n b_i^2)}} \quad (5)$$

If the normalized correlation $N(\mathbf{y}, \mathbf{w})$ exceeds some threshold, we declare that the watermark is present; otherwise, the detector will pick the next flow from the IPD database and try again. Note that only recent flows need to be stored in the database, based on the total expected delay in the network. We next present an analysis of the error rates expected from the RAINBOW scheme.

4 System Analysis

To analyze the performance of RAINBOW, we must first define a model for the network traffic and delay. We will model network flows as independent Poisson processes, thus the IPDs will be distributed exponentially. We will model network delays as i.i.d. exponential as well, which implies that the jitter (difference of two delays) is i.i.d. according to a zero-mean Laplace distribution denoted by $Lap(0, b_\delta)$, where $2b_\delta^2$ is the variance of the jitter. Of course, in a real network, delays will have some correlation; we compare the PDF of real observed jitter on a connection over PlanetLab [1] with a best-fit Laplace distribution in Figure 3. We can see that the real PDF has greater support at 0, and the Laplace distribution has a heavier tail. This means that our analysis of error rates will be conservative, since 0 jitter will result in no error for our detection scheme. We have also conducted similar experiments with the same results on Tor anonymous network [7] to consider the other application of watermarking.

4.1 Hypothesis Testing

We use *hypothesis testing* [15] to analyze performance of the RAINBOW for the normalized correlation detector

[1] Throughout this paper, by attacker we mean the attacker to the watermarking scheme.

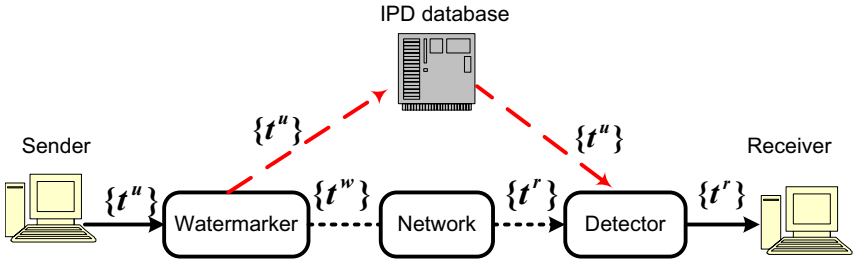


Figure 2. Model of RAINBOW network flow watermarking system.

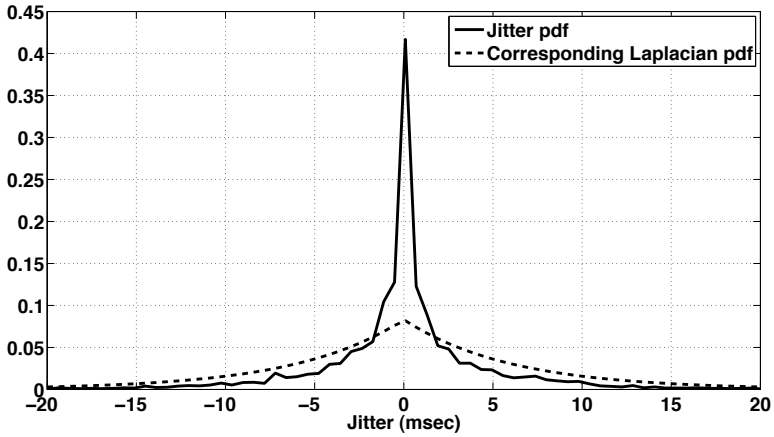


Figure 3. A comparison of observed jitter and a fitted Laplace distribution.

of (5). For ease of presentation, we start by analyzing a correlation decoder and then derive the error probability of the normalized correlation detector.

Let us define the *test statistic* to be the correlation between subtracted IPD, $\mathbf{y} = \tau^{\mathbf{r}} - \tau^{\mathbf{u}}$, and the watermark sequence, $\mathbf{w}$, as:

$$T[\mathbf{y}] = \langle \mathbf{y}, \mathbf{w} \rangle = \Sigma y_i w_i \tag{6}$$

We aim to distinguish between two different hypotheses:

- H_0 (*null hypothesis*): $\tau^{\mathbf{r}}$ is a new, unwatermarked flow, unlinked to $\tau^{\mathbf{u}}$, *and*
- H_1: $\tau^{\mathbf{r}}$ is the result of a watermarked flow $\tau^{\mathbf{u}}$ passing through the network.[2]

Under hypothesis H_0, the received IPD is $\tau^{\mathbf{r}} = \tau^{\mathbf{u}^*} + \delta$, for some unrelated flow $\tau^{\mathbf{u}^*}$. After subtracting the flow of interest $\tau^{\mathbf{u}}$, we have

$$\begin{cases} y_{|H_0} = (\tau^{\mathbf{u}^*} + \delta) - \tau^{\mathbf{u}} = (\tau^{\mathbf{u}^*} - \tau^{\mathbf{u}}) + \delta \\ y_{|H_1} = \mathbf{w} + \delta \end{cases} \tag{7}$$

The corresponding correlation test statistics of (6) under the two hypotheses are

$$\begin{cases} T[y_{|H_0}] = \langle \tau^{\mathbf{u}^*} - \tau^{\mathbf{u}}, \mathbf{w} \rangle + \langle \delta, \mathbf{w} \rangle \\ T[y_{|H_1}] = \langle \delta, \mathbf{w} \rangle + \langle \mathbf{w}, \mathbf{w} \rangle = \langle \delta, \mathbf{w} \rangle + na^2 \end{cases} \tag{8}$$

where n is the number of packets in the observed flows and a is the watermark absolute amplitude as defined in (1).

4.2 False Errors

The decision rule for detecting watermarks uses a threshold η, such that if $T[y] \geq \eta$, the watermark is said to be present, and absent otherwise. We can therefore express the false positive and false negative rates in terms of η:

$$P_{\text{FP}} = P(T[y_{|H_0}] \geq \eta) \tag{9}$$
$$P_{\text{FN}} = P(T[y_{|H_1}] < \eta) \tag{10}$$

[2]Note that there is another possibility, namely that $\tau^{\mathbf{r}}$ is a *watermarked* flow, but not corresponding to $\tau^{\mathbf{u}}$. However, we ignore this case because errors in this scenario do not matter: if the flow is said to be watermarked, then the detection algorithm is correct, and if it is said to be unwatermarked, it will later be tested against the correct $\tau^{\mathbf{u}}$.

Before analyzing the false error rates, we present the following lemma:

Lemma 1. *If X_1 and X_2 are two independent random variables distributed according to $X_1 \sim Lap(0, b_1)$ and $X_2 \sim Lap(0, b_2)$, the tail of the distribution of $X_1 + X_2$ can be approximated by $Lap(0, \sqrt{b_1^2 + b_2^2})$.*

Proof. The characteristics function of $Lap(0, b)$ is $\Phi(t) = \frac{1}{1+b^2s^2}$. Since X_1 and X_2 are independent, the characteristic function of the distribution of $X_1 + X_2$ is:

$$\frac{1}{1 + b_1^2 s^2} \cdot \frac{1}{1 + b_2^2 s^2} = \frac{1}{1 + (b_1^2 + b_2^2)s^2 + b_1^2 b_2^2 s^4} \quad (11)$$

For small s (which corresponds to the tail of distribution), s^4 becomes vanishingly small with respect to other terms, and so the characteristic function approaches that of $Lap(0, \sqrt{b_1^2 + b_2^2})$. □

Corollary 1. *Suppose $X_1, \ldots, X_n$ are i.i.d distributed with $Lap(0, b)$. For $Y = \sum_{i=1}^{n} X_i$, the tail of Y's distribution can be approximated by $\widetilde{Y} \sim Lap(0, \sqrt{n}b)$.*

Given hypothesis H_1 is true, from (8) we have

$$\begin{aligned} T[y_{|H_1}] &= \langle \delta, \mathbf{w} \rangle + na^2 && (12)\\ &= \sum_{i=1}^{n} w_i \delta_i + na^2 && (13)\\ &= \underbrace{\sum_{i=1}^{n_1} (a\delta_i)}_{(*)} + \underbrace{\sum_{i=n_1+1}^{n} (-a\delta_i)}_{(**)} + na^2 && (14) \end{aligned}$$

where n_1 is the number of w_i's that are positive (We can assume without loss of generality that w_i's are sorted, since δ_i's are i.i.d.). Since the Laplace distribution is symmetric, both δ_i and $-\delta_i$ are distributed as $Lap(0, b_\delta)$. We apply the Corollary 1 to the terms (*) and (**) separately and then to the resulting terms to obtain:

$$\langle \delta, \mathbf{w} \rangle \sim Lap(0, \sqrt{n}ab_\delta) \quad (15)$$

Therefore:

$$T[y_{|H_1}] \sim Lap(na^2, \sqrt{n}ab_\delta) \quad (16)$$

For H_0, we need to consider two terms: $\langle \tau^{\mathbf{u}^*} - \tau^{\mathbf{u}}, \mathbf{w} \rangle$ and $\langle \delta, \mathbf{w} \rangle$. For the first term, recall that τ^{u^*} and τ^u are IPDs from two independent exponential distributions. If the rates of the corresponding Poisson processes are both equal to λ (which is the worst case scenario for false errors) $\tau_i^{u^*} - \tau_i^u$ will be distributed according to $Lap(0, 1/\lambda)$. By applying a similar analysis to above, we have that:

$$\langle \tau^{\mathbf{u}^*} - \tau^{\mathbf{u}}, \mathbf{w} \rangle \sim Lap(0, \frac{\sqrt{n}a}{\lambda}) \quad (17)$$

Combining (15) and (17), and applying the lemma, we have:

$$T[y_{|H_0}] \sim Lap(0, a\sqrt{n(b_\delta^2 + 1/\lambda^2)}) \quad (18)$$

Normalized Correlation: Normalized correlation will divide the above test statistic by the norms of the two distributions, i.e.:

$$\tilde{T}[\mathbf{y}] = \frac{T[\mathbf{y}]}{||\mathbf{y}|| \cdot ||\mathbf{w}||} \quad (19)$$

We know that $||\mathbf{w}|| = \sqrt{n}a$. For large n, we can approximate $||\mathbf{y}||$ by $\sqrt{nE(y_i^2)}$ by the law of large numbers. Taking $\lambda = \lambda^*$ as before, we have:

$$\begin{aligned} E(y_i^2) &= 2(b_\delta^2 + 1/\lambda^2) \quad \text{under } H_0 && (20)\\ E(y_i^2) &= 2b_\delta^2 \quad \text{under } H_1 && (21) \end{aligned}$$

Therefore:

$$\begin{aligned} \tilde{T}[y_{|H_0}] &= \frac{Lap(0, a\sqrt{n(b_\delta^2 + 1/\lambda^2)})}{\sqrt{n}a \cdot \sqrt{2n(b_\delta^2 + 1/\lambda^2)}} && (22)\\ &= Lap(0, \frac{1}{\sqrt{2n}}) && (23) \end{aligned}$$

$$\begin{aligned} \tilde{T}[y_{|H_1}] &= \frac{Lap(na^2, \sqrt{n}ab_\delta)}{\sqrt{n}a \cdot \sqrt{2n}b_\delta} && (24)\\ &= Lap(\frac{a}{\sqrt{2}b_\delta}, \frac{1}{\sqrt{2n}}) && (25)\\ &= Lap(\gamma, \frac{1}{\sqrt{2n}}) && (26) \end{aligned}$$

where γ is the ratio of watermark amplitude to jitter standard deviation:

$$\gamma = a/\sqrt{2}b_\delta, \quad (27)$$

Recall that the cumulative distribution function for the Laplace distribution $Lap(\mu, b)$ is:

$$F(x) = \begin{cases} \frac{1}{2}e^{-\frac{\mu - x}{b}} & \text{if } x < \mu \\ 1 - \frac{1}{2}e^{-\frac{x - \mu}{b}} & \text{if } x \geq \mu \end{cases} \quad (28)$$

If we let $F_0(x)$ and $F_1(x)$ be the cumulative distribution functions of the two Laplace distributions above, we will have that $\widetilde{P_{\text{FP}}} = 1 - F_0(\widetilde{\eta})$ and $\widetilde{P_{\text{FN}}} = F_1(\widetilde{\eta})$, where $\widetilde{\eta}$ is the decision threshold used in the detector. Note that in practice, we want to set $0 < \widetilde{\eta} < \gamma$, since outside that range either false positive or false negative rate will be at least 1/2. Therefore, we have:

$$\begin{aligned} \widetilde{P_{\text{FP}}} &= 1 - \left(1 - \frac{1}{2}e^{-\widetilde{\eta}\sqrt{2n}}\right) && (29)\\ &= \frac{1}{2}e^{-\widetilde{\eta}\sqrt{2n}} && (30) \end{aligned}$$

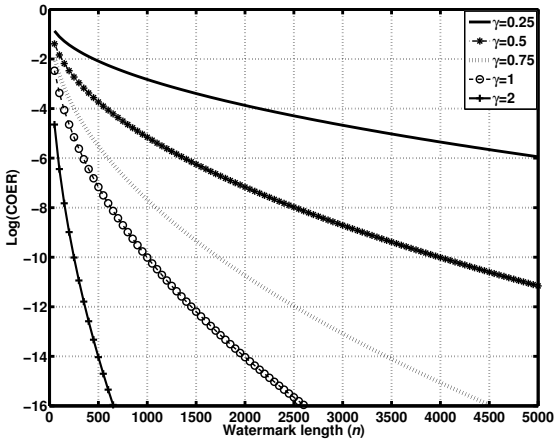


Figure 4. Cross-over error rate for different γ and n.

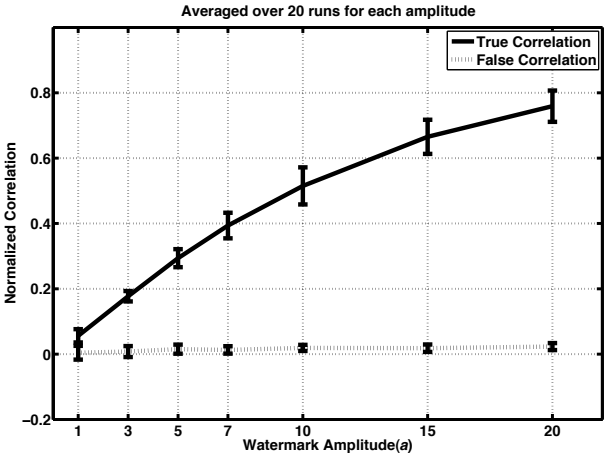


Figure 6. Normalized correlation test statistic for different watermark amplitudes.

$$\widetilde{P_{\rm FN}} = \frac{1}{2}e^{-(\gamma-\widetilde{\eta})\sqrt{2n}} \quad (31)$$

$$= \frac{1}{2}e^{(\widetilde{\eta}-\gamma)\sqrt{2n}} \quad (32)$$

4.3 Discussion

The error rates are a function of the number of packets, n, and the ratio of watermark amplitude to jitter, γ of (27), which can be thought of as a signal-to-noise ratio (SNR). To remain invisible, we must use low-powered watermarks, reducing γ, and therefore we need to compensate by increasing n.

This tradeoff can be seen in Figure 4. We plot the error rate for different choices of γ and n. Since by changing $\widetilde{\eta}$ we can trade off false positives for false negatives, we measure the cross-over error rate (COER), which is the point where $P_{\rm FN} = P_{\rm FP}$. The important observation to be drawn is that, even for $\gamma = 0.5$, low error rates can be achieved with fewer than 1000 packets. This small value of γ promises tiny watermark amplitudes as small as 5 milliseconds which are highly invisible, as visibility experiments show in Section 7. We also sketch the probability of false negative when the false positive is fixed to 10^{-3} and 10^{-6} for different choices of γ and n in Figure 5. In some scenarios, we can sacrifice false positive for false negative to improve the watermarking system efficiency. As an example, we can achieve false positive of 10^{-3} and false negative of 10^{-6} for SNR ratio of $\gamma = 1$ and $n = 200$ number of watermarked packets (Figure 5(a)).

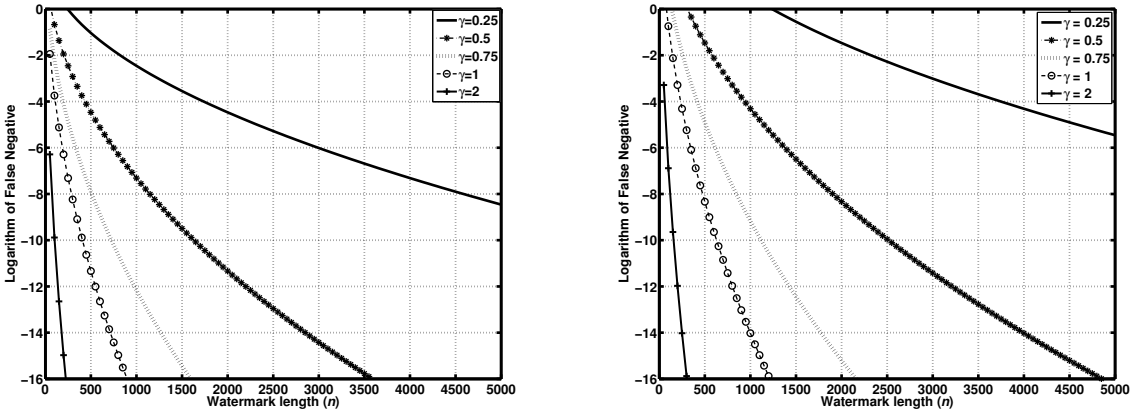
Note that in our analysis, b_δ models the jitter *within* the enterprise network. However, to be unnoticeable to the attacker, the watermark amplitude needs to be small relative to the jitter observed by the attacker, which includes the jitter in the Internet connection from the attacker to the enterprise. This will tend to be considerably larger, and so we expect that installations may be able to use γ values of 2 or more, resulting in extremely efficient detection.

5 Implementation Results

We implemented the watermarking scheme and tested it by using replayed SSH connections, using timings collected from real traces at the North Carolina State University, as well as at the University of Illinois. Our tests were carried out over the PlanetLab infrastructure.

In the first experiment, we watermarked SSH flows between two specific nodes for different values of watermark amplitude ($1, 3, 5, 7, 10, 20 msec$). We show the test statics for both true correlation (hypothesis one) and false correlation (hypothesis zero), along with their standard deviations in Figure 6 (each experiment is run for 20 times and the average jitter standard deviation over the link is about $\delta_b = 10 msec$). As we expect from analysis, false detection metric has a mean of around zero, and a variance steadily constant (because n is fixed). For hypothesis one, the statistic mean increases linearly with watermark amplitude (recall that mean of true correlation is $\frac{a}{\sqrt{2}b_\delta}$), and variance shows not much change for different experiments.

In the second experiment we watermarked 100 SSH flows of length $N = 5000$ packets with fixed watermarked amplitude of $a = 10ms$ between two specific nodes (and also the same watermark bits). The high number of flows helps to measure the variance of metrics with more confidence. Figure 7(a) shows true detection metric, and false



(a) False Positive = 10^{-3}

(b) False Positive = 10^{-6}

Figure 5. False Negative error for different values of γ and n (False).

detection metric along with their standard deviation for different number of packets n. Mean of true correlation does not vary with n because watermark amplitude is fixed and network jitter does not vary that much; mean of false correlation is almost zero as we expect from analysis. Standard deviation of true correlation shows to vary with $1/\sqrt{n}$ as we expect from analysis. Fortunately, false correlation shows a slightly smaller standard deviation which results in even fewer false positives. This is because we considered the worst case in analysis, i.e., equal rate unwatermarked flows. Figure 7(b) shows the COER estimated by fitting the errors rates to a Laplace distribution; comparing with Figure 4 for $\gamma = 1$ this experimental COER highly matches the analytical results. Based on this, we can achieve the tiny COER of 10^{-6} with fewer than 400 packets, which means that a typical SSH connection can be classified as a stepping stone or not within about 3 minutes. Similar passive and watermarking schemes require much more time to achieve similar error rates.

Comparing error rates of RAINBOW with those of previous passive schemes and blind watermarking schemes, RAINBOW outperforms them by orders of magnitude. The passive scheme of [20] which uses similar correlation mechanisms as RAINBOW, achieves false errors of 10^{-2} for different parameters. IPD-based watermarking scheme of [21] achieves false negative rates of 10^{-2} and false positive rates of at most 10^{-5}. These are far worse than what RAINBOW achieves.

5.1 Resource constraints

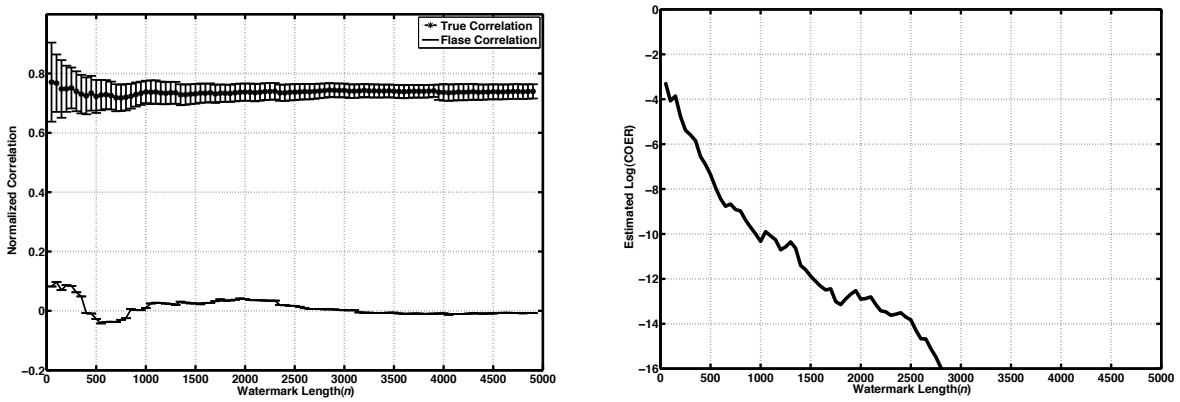
In this section we evaluate the required resources for RAINBOW in the case of stepping stone configurations. Of course, the resources required will be dependent on the number of low-rate connections, which in turn will depend on the size of the organization. We estimate the parameters needed to detect stepping stones in an organization such as the Coordinated Science Laboratory (CSL) at the University of Illinois at Urbana–Champaign. CSL has about 400 members, so we will assume as a worst-case that each member is performing a low-rate connection from the outside. Using a C++ implementation of RAINBOW, running on a 1.6 GHz Linux server with 1 GB of RAM, we can perform selective correlation (a more resource-intensive method that is robust to packet deletions and insertions, discussed in the next section) with 400 flows, using a watermark length of $n = 5000$, in 0.4μs. Table 1 lists the storage requirements for the IPD table for various choices of n.

Given the small size of the CPU and memory constraints, and the fact that they scale linearly with the number of flows, it is easy to see that much larger organizations can be supported using a commodity PC. For extremely large organizations, stepping stone detection can be partitioned among routers within sub-networks; e.g., in an organization such as the University of Illinois, each department can run its own stepping stone detection.

The choice of n presents a tradeoff between detection accuracy, watermark amplitude, and resource constraints. However, as we saw earlier, RAINBOW is effective with only a few hundred packets, whereas other passive schemes require many more packets [17, 24, 8, 20, 3], hence the resource constraints of RAINBOW will be significantly lower.

6 Selective Correlation

In the previous sections we analyzed the performance of the detector based on normalized correlation. In our anal-



(a) Normalized correlation metric

(b) Estimated COER

Figure 7. Experimental detection performance for different watermark lengths.

Table 1. Maximum memory usage of the RAINBOW watermarking system for a medium-size network.

n parameter	Memory (MB)
50	0.15
100	0.3
200	0.6
500	1.5
1000	3.1

ysis and implementation, we assumed that there is a one-to-one relation between packets of watermarked flow and received flow; i.e., no packets are added to or removed from the flow between watermark insertion and watermark detection. This is often not the case, however, as real-world implementations introduce several causes for packets removal and insertion. For example, retransmissions at the TCP layer will introduce packets into one of the streams.[1] Applications may also repacketize flows while relaying them. Setup packets, such as TCP SYN/ACK packets and packets sent to initialize an SSH connection will also show up in only one of the two flows.

So, a practical watermark detector should be robust to packet addition and removal, i.e., work efficiently despite them. Among existing work, only recent schemes have considered repacketization and other natural perturbations [16, 23], while other work has looked at the presence of adversarial packet insertion and removal, or *chaff* [8, 3, 19]. Our normalized correlation scheme analyzed thus far is fragile to packet addition and removal, but with a modification we call *Selective Correlation* it shows promising performance

[1]Though proper parsing of the TCP packets can be used to detect such retransmissions and remove them from consideration.

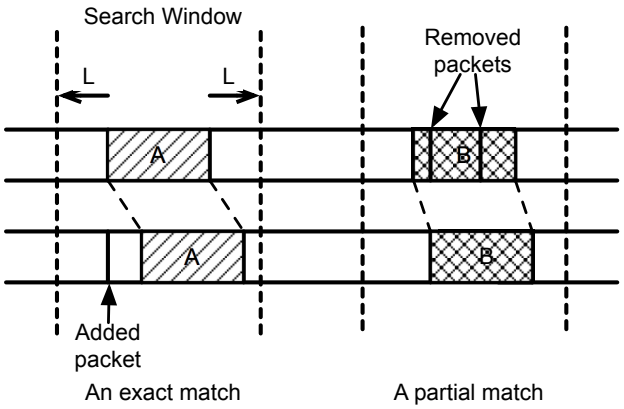
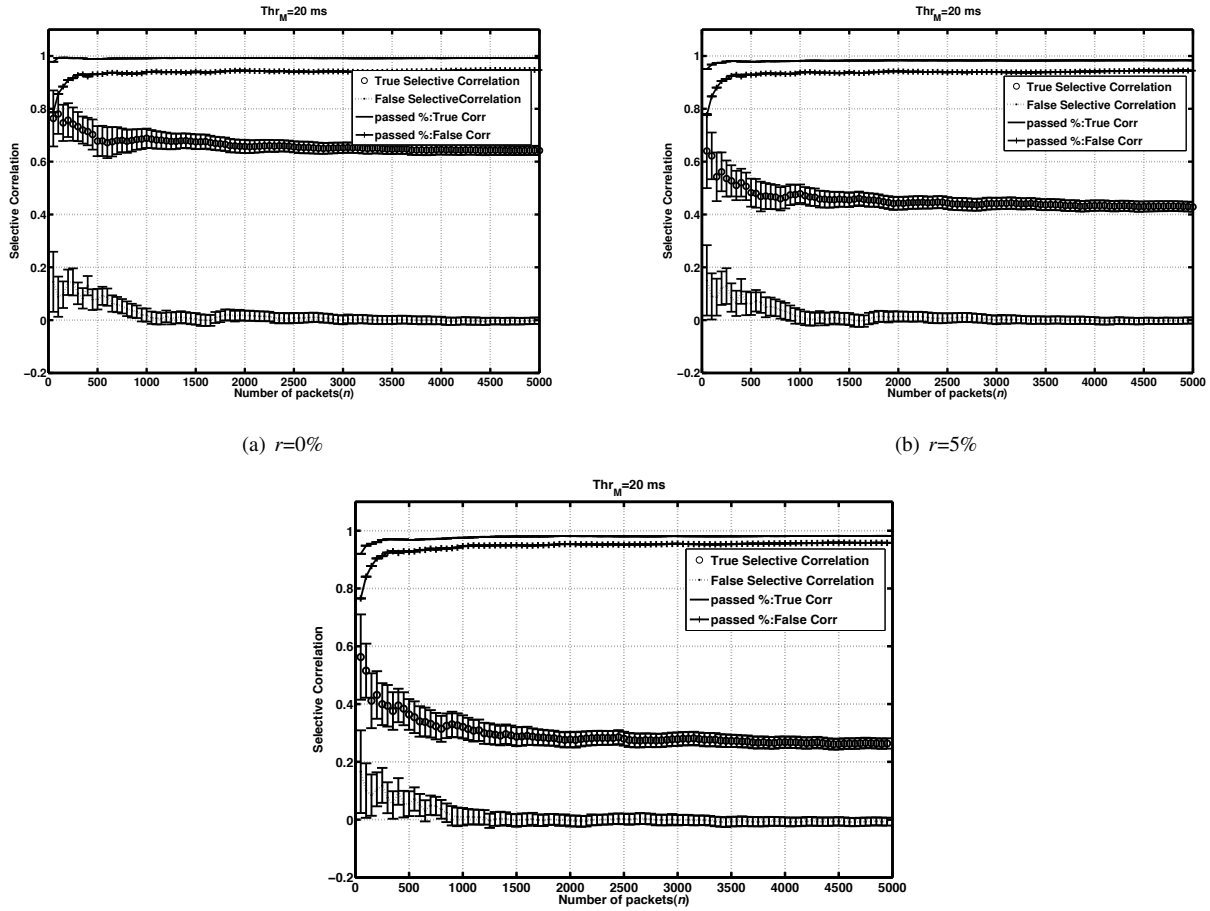


Figure 8. Selective Correlation

dealing with packet addition and removal carried out at a relatively high rate.

Selective Correlation scheme: For selective correlation, we add a *matching step* to the detector, which will pre-process $\tau^{\mathbf{r}}$, $\tau^{\mathbf{r}}$, and $\mathbf{w}$, before they are passed to the normalized correlation step. The aim of this step is to find and remove packets that do not have a corresponding match in the other flow.

The main idea is to use sliding windows to match IPD values of one flow by those of the other flow. Figure 8 illustrates how the matching step works. For any IPD value of the received flow, τ_i^r, if the absolute difference from the corresponding IPD in database, τ_j^u, is smaller than η_M, packets are passed through as matched, along with the corresponding watermark bit. If not, the matching block tries to find an IPD in a $[j-L, j+L]$ window of $\tau^{\mathbf{u}}$ with the smallest IPD difference from τ_i^r that is also smaller than η_M. If no match is found the packet is dropped. L is the maximum



(a) r=0%

(b) r=5%

(c) r=10%

Figure 9. Selective correlation performance for different ratio of add/drop packets (r).

expected change in number of packets and η_M depends on jitter variance.

To account for the case where too many packets are not matched, the detector also monitors the percentage of matched packets and declares the received flow as unwatermarked if this number is smaller than a threshold η_R.

Implementation results: We implemented selective correlation scheme over the same watermarked connections in PlanetLab, after adding and removing different percentages of packets to the flows. We set η_M to be twice the average standard deviation of jitter; i.e., $\eta_M = 20ms$, and L twice the maximum number of packets expected to be added or removed. Figure 9 illustrates true selective correlation and false selective correlation along with the percentage of packets matched in each case. If percentage of matched packets falls bellow some reasonable threshold, η_R, detector decides flow to be not watermarked.

For the case that we do not have packet count changes (Figure 9(a)), selective correlation outperforms the simple correlation scheme (Figure 7(a)). This is because selective correlation removes IPDs with high jitter added. As fraction of packets added and/or removed increases, mean of true selective correlation decreases as shown in Figures 9(b) and 9(c). This leads detection performance to decrease, but even for 20 percent of packets changed (10% added and 10% removed), detection can be performed efficiently.

7 Watermark Invisibility

An efficient network flow watermarking scheme needs to be invisible to prevent the watermark from being detected and possibly removed by an active attacker. This also prevents the watermark from interfering with normal users traffic. Because of embedding large amplitude watermarks, previous flow watermarking schemes are not invisible; several interval-based watermarking schemes [19, 16, 23] have shown to be subject to detection and removal [11] (it should be mentioned that changing some watermarking parame-

ters, e.g. interval length, in these schemes from the original values in the corresponfing papers improves invisibility, but drastically ruins false detection errors which make the schemes practically useless). Peng et al. [13] show how the *Kolmogrov–Smirnov test* (K–S test) is efficient in detecting large amplitude *QIM* watermarks applied to inter-packet delays. We use the Kolmogorov–Smirnov test to discuss invisibility of RAINBOW flow watermarking scheme.

The K-S test is used to determine whether two samples from a sequence of two observations (or one observation and samples drawn from a references probability distribution function) belong to the same distribution by measuring the maximum distance between empirical distribution functions (or the empirical distribution function and the reference distribution function). In case of a given reference distribution function $F(x)$, the value of the K–S test is:

$$\sup_x |F_n(x) - F(x)|,$$

where $F_n(x)$ denotes the empirical distribution function from a sample of n observations.

In the first experiment we ran the K–S test against the non-watermarked and watermarked version of a SSH flow, transmitted in the same network (with similar network delay). The average K–S distance between them (averaged over 10 connections) is 0.0082 which results in a 98% confidence in declaring them to be from the same distribution. In other words, watermark presence on the flows would be transparent to normal users and (limited) attackers.

In the second experiment we considered a more intelligent/powerful attacker who sends a flow to the watermarker and receives it on another compromised host. Since the attacker has the original flow, he only needs to discriminate between $w + \delta_i$ and δ_j , where δ_i and δ_j are different jitters (measured over PlanetLab). We compared two scenarios: K–S test between $w+\delta_1$ and δ_2 and K–S test between δ_3 and δ_4. Figure 10 shows the difference between K–S statistics in the two different scenarios for different values of γ. As γ decreases, the attacker loses his chance to distinguish between watermarked and unwatermarked flows. Comparing with results of Section 4, we see that there is a tradeoff between different watermarking attributes, i.e., invisibility and robustness. A similar K–S experiment on other flow watermarking schemes returns much higher differences, which makes them suspect to attacks [11].

Gianvecchio et al. use information theory tools to invent new metrics for efficient detection of covert timing channels [9]. We use their entropy-based tools, *EN* and *CCE* tests, on a number of watermarked SSH flows (each 5000 packets) and their corresponding unwatermarked (but jittered) flows. Table 2 shows the averaged test metrics for regular (unwatermarked) and watermarked SSH flows. As results show, even for large values of γ, watermarking does not change *EN* and *CCE* test results significantly (the decision thresholds for *EN* and *CCE* tests are 21.20 and 2.17, respectively). This shows that RAINBOW remains invisible in the face of these information-theoretical tools.

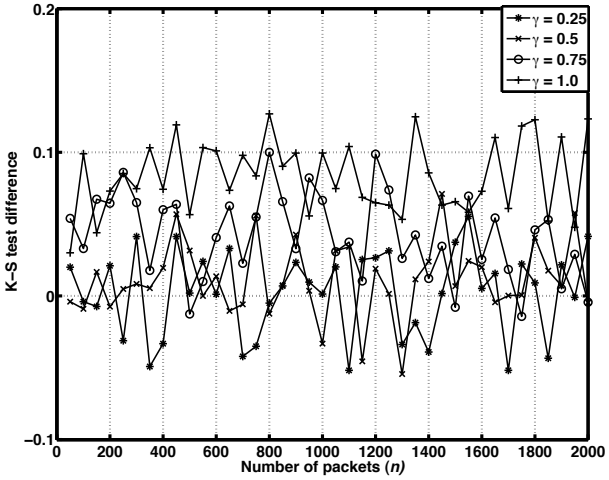


Figure 10. Kolmogorov–Smirnov test difference

8 Conclusions and Future Research

We proposed a novel non-blind network flow watermarking scheme called RAINBOW, for linking flows. RAINBOW combines some of the advantages of passive traffic analysis with watermarking schemes. Like passive traffic analysis, RAINBOW does not interfere with regular users by inserting large delays that are used in existing watermarking schemes; in fact, we show that RAINBOW is *invisible* to detection by an attacker. Like other watermarking schemes, RAINBOW achieves very low false error rates. In fact, we show, both through analysis and by means of experiment, that the false error rates of RAINBOW are orders of magnitude lower for short observation periods than existing passive and active schemes. RAINBOW can also be made robust to high rates of packet addition and removal by introducing selective correlation, at the cost of somewhat increased observation period lengths. In our future work, we intend to explore coding tools to increase the efficiency of RAINBOW and explore the possibility of a blind or semi-blind watermark scheme that remains invisible.

Acknowledgments

We would like to thank the anonymous reviewers and our shepherd, Virgil Gligor, for helpful suggestions on an earlier version of the paper. This work was supported in part by

Table 2. Entropy test results to evaluate invisibility of RAINBOW.

γ	EN test		CCE test	
	Regular	Watermarked	Regular	Watermarked
0.25	13.7191	13.7499	2.2475	2.2514
0.50	13.8061	13.7661	2.2476	2.2493
0.75	13.7651	13.7590	2.2471	2.2526
1.00	13.7711	13.7903	2.2484	2.2498
2.00	13.7545	13.6533	2.2496	2.2498

the National Science Foundation awards CNS–062761 and CNS–0831488.

References

[1] A. Bavier, M. Bowman, B. Chun, D. Culler, S. Karlin, S. Muir, L. Peterson, T. Roscoe, T. Spalink, and M. Wawrzoniak. Operating systems support for planetary-scale network services. In R. Morris and S. Savage, editors, *Symposium on Networked Systems Design and Implementation*, pages 253–266. USENIX, Mar. 2004.

[2] W. Bender, D. Gruhl, N. Morimoto, and A.Lu. Techniques for data hiding. *IBM Systems Journal*, 35(3/4):313–336, 1996.

[3] A. Blum, D. X. Song, and S. Venkataraman. Detection of interactive stepping stones: Algorithms and confidence bounds. In E. Jonsson, A. Valdes, and M. Almgren, editors, *International Symposium on Recent Advances in Intrusion Detection*, volume 3224 of *Lecture Notes in Computer Science*, pages 258–277. Springer, Sept. 2004.

[4] B. Chen and G. W. Wornell. Quantization index modulation methods for digital watermarking and information embedding of multimedia. *The Journal of VLSI Signal Processing*, 27(1–2):7–33, 2001.

[5] I. Cox, J. Kilian, T. Leighton, and T. Shamoon. Secure spread spectrum watermarking for multimedia. *IEEE Transactions on Image Processing*, 6(12):1673–1687, 1997.

[6] G. Danezis. The traffic analysis of continuous-time mixes. In D. Martin and A. Serjantov, editors, *Workshop on Privacy Enhancing Technologies*, volume 3424 of *Lecture Notes in Computer Science*, pages 35–50. Springer, May 2004.

[7] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In M. Blaze, editor, *USENIX Security Symposium*, Berkeley, CA, USA, 2004. USENIX Association.

[8] D. Donoho, A. Flesia, U. Shankar, V. Paxson, J. Coit, and S. Staniford. Multiscale stepping-stone detection: detecting pairs of jittered interactive streams by exploiting maximum tolerable delay. In A. Wespi, G. Vigna, and L. Deri, editors, *International Symposium on Recent Advances in Intrusion Detection*, volume 2516 of *Lecture Notes in Computer Science*, pages 16–18. Springer, Oct. 2002.

[9] S. Gianvecchio and H. Wang. Detecting covert timing channels: an entropy-based approach. In P. Ning, S. D. C. di Vimercati, and P. F. Syverson, editors, *ACM Conference on Computer and Communications Security*, pages 307–316. ACM, 2007.

[10] T. He and L. Tong. Detecting encrypted stepping-stone connections. *IEEE Transactions on Signal Processing*, 55(5):1612–1623, May 2007.

[11] N. Kiyavash, A. Houmansadr, and N. Borisov. Multi-flow attacks against network flow watermarking schemes. In P. van Oorschot, editor, *USENIX Security Symposium*, Berkeley, CA, USA, 2008. USENIX Association.

[12] B. N. Levine, M. K. Reiter, C. Wang, and M. Wright. Timing attacks in low-latency mix systems. In A. Juels, editor, *Financial Cryptography*, volume 3110 of *Lecture Notes in Computer Science*, pages 251–265. Springer, Feb. 2004.

[13] P. Peng, P. Ning, and D. S. Reeves. On the secrecy of timing-based active watermarking trace-back techniques. In V. Paxson and B. Pfitzmann, editors, *IEEE Symposium on Security and Privacy*, pages 334–349. IEEE Computer Society Press, May 2006.

[14] B. Pfitzmann and P. McDaniel, editors. *IEEE Symposium on Security and Privacy*, May 2007.

[15] H. V. Poor. *An Introduction to Signal Detection and Estimation*. Springer-Verlag, 1998.

[16] Y. Pyun, Y. Park, X. Wang, D. S. Reeves, and P. Ning. Tracing traffic through intermediate hosts that repacketize flows. In G. Kesidis, E. Modiano, and R. Srikant, editors, *IEEE Conference on Computer Communications (INFOCOM)*, pages 634–642, May 2007.

[17] S. Staniford-Chen and L. T. Heberlein. Holding intruders accountable on the Internet. In C. Meadows and J. McHugh, editors, *IEEE Symposium on Security and Privacy*, pages 39–49. IEEE Computer Society Press, May 1995.

[18] X. Wang, S. Chen, and S. Jajodia. Tracking anonymous peer-to-peer VoIP calls on the Internet. In C. Meadows, editor, *ACM Conference on Computer and Communications Security*, pages 81–91, New York, NY, USA, Nov. 2005. ACM.

[19] X. Wang, S. Chen, and S. Jajodia. Network flow watermarking attack on low-latency anonymous communication systems. In Pfitzmann and McDaniel [14], pages 116–130.

[20] X. Wang, D. Reeves, and S. F. Wu. Inter-packet delay based correlation for tracing encrypted connections through stepping stones. In D. Gollmann, G. Karjoth, and M. Waidner, editors, *European Symposium on Research in Computer Security*, volume 2502 of *Lecture Notes in Computer Science*, pages 244–263. Springer, Oct. 2002.

[21] X. Wang and D. S. Reeves. Robust correlation of encrypted attack traffic through stepping stones by manipulation of interpacket delays. In V. Atluri, editor, *ACM Conference on Computer and Communications Security*, pages 20–29, New York, NY, USA, 2003. ACM.

[22] T. Ylonen and C. Lonvick. The secure shell (SSH) protocol architecture. RFC 4251, Jan. 2006.

[23] W. Yu, X. Fu, S. Graham, D.Xuan, and W. Zhao. DSSS-based flow marking technique for invisible traceback. In Pfitzmann and McDaniel [14], pages 18–32.

[24] Y. Zhang and V. Paxson. Detecting stepping stones. In S. Bellovin and G. Rose, editors, *USENIX Security Symposium*, pages 171–184, Berkeley, CA, USA, Aug. 2000. USENIX Association.