SWIRL: A Scalable Watermark to Detect Correlated Network Flows

Amir Houmansadr Nikita Borisov

Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
E-mail: {ahouman2,nikita}@illinois.edu

Abstract

Flow watermarks are active traffic analysis techniques that help establish a causal connection between two network flows by content-independent manipulations, e.g., altering packet timings. Watermarks provide a much more scalable approach for flow correlation than passive traffic analysis. Previous designs of scalable watermarks, however, were subject to multi-flow attacks. They also introduced delays too large to be used in most environments.

We design SWIRL, a Scalable Watermark that is Invisible and Resilient to packet Losses. SWIRL is the first watermark that is practical to use for large-scale traffic analysis. SWIRL uses a flow-dependent approach to resist multi-flow attacks, marking each flow with a different pattern. SWIRL is robust to packet losses and network jitter, yet it introduces only small delays that are invisible to both benign users and determined adversaries. We analyze the performance of SWIRL both analytically and on the PlanetLab testbed, demonstrating very low error rates. We consider applications of SWIRL to stepping stone detection and linking anonymous communication. We also propose a novel application of watermarks to defend against congestion attacks on Tor.

1 Introduction

Network intruders usually try to hide their real location by relaying their traffic through a number of intermediate hosts, called *stepping stones* [25]. The traffic is encrypted, preventing simple identification of stepping stones; instead, traffic analysis techniques are used to find flows that have similar characteristics, based on features such as packet timings, counts, and sizes [3, 6, 17, 20, 23, 25]. Traffic analysis can also be used to attack anonymous communication systems by finding relationships between two flows that would otherwise be unlinkable [1, 4, 16].

Traditionally, traffic analysis is performed as a passive attack, by observing the candidate flows and trying to find the flow relations through different statistical analysis [1, 3, 4, 6, 16, 17, 20, 23, 25]. More recently, network flow watermarks have been proposed as an *active* alternative to perform traffic analysis more efficiently. Watermarks are more *scalable*, as they require asymptotically less communication and computation; they also can operate on shorter flows and provide lower error rates than passive analysis.

Previous watermark designs can be divided into two categories: packet-based watermarks that operate on individual delays between packets [21], and interval-based watermarks that perform an operation on an entire interval [15, 19, 24]. The former category is not robust to packet losses, reorderings, and insertions; the latter are subject to a multi-flow attack [10] that can recover and remove the watermark. In addition, these watermarks introduce large delays, making them not suitable for practical applications.[1]

We introduce SWIRL, a Scalable Watermark that is Invisible and Resilient to packet Losses. SWIRL is an interval-based watermark, but it uses a novel approach to resist multi-flow attacks. The watermark pattern is chosen based on the characteristics of the flow being marked; as a result, each flow is marked with a different pattern. SWIRL watermarks introduce small delays to the network flows, and thus are practical to deploy in real-world scenarios. The small amount of distortion also makes SWIRL invisible to state-of-the-art information-theoretic tools for covert channel detection [8].

We perform a mathematical analysis of the error rates of SWIRL, showing that it can achieve very low false error rates on short flows. We validate our analysis against simulations and an implementation running on the PlanetLab testbed [2]. We show experimentally that SWIRL is resistant to the multi-flow attacks.

SWIRL provides the first practical approach to large scale traffic analysis; it therefore extends the reach of traffic analysis attacks in both anonymous systems and network attack attribution. We also consider a novel application

[1] RAINBOW [9] is a packet-based watermark that *is* robust to packet losses; however, it is designed to supplement, rather than replace, passive traffic analysis and thus does not scale to large scenarios.

of watermarks to defend against a congestion attack in the Tor anonymizing network [5]. We show that a watermark, normally a privacy-invasive tool used to link anonymous flows, can actually help protect users' privacy by preventing attackers from creating routing loops. The properties of SWIRL provide a practical defense where previous traffic analysis approaches would not be appropriate.

The rest of this paper is organized as follows. We review some background on network flow watermarking in Section 2. In Section 3 we describe the design of the SWIRL watermarking scheme. We analyze SWIRL by modeling the network flow behavior in Section 4 to provide false errors analysis of the scheme. We evaluate SWIRL with simulations as well as implementation in Section 5. We discuss watermark invisibility and resilience to multi-flow attacks in Section 6. In Section 7 we provide a novel application of SWIRL to Tor congestion attacks . Finally, we conclude in Section 8.

2 Background

We first review the watermarking setting. Watermarks provide a content-independent way to tag traffic so that correlated flows can later be recognized. Figure 1 shows the general model of a network flow watermarking scheme. A network flow passing through a *watermarker* gets *watermarked* by changing the timing information of packets, i.e., applying specific delays on the packets. The flow then travels along a noisy channel, which may include various networks, stepping stones, and anonymizing systems. This channel introduces further delays, and might also drop, reorder, or duplicate packets or repacketize the flow. After the channel, the flow arrives at the watermark *detector*, which inspects it for the inserted watermark pattern. The pattern encoding is based upon a secret watermark key, shared between the watermarker and detector. In a *blind* scheme, this is the only information the watermarker and detector exchange; in a *non-blind* scheme, the watermarker also sends information about the watermarked flow to the detector through an out-of-band channel.

Watermarking has several advantages over passive traffic analysis schemes. First, watermarks can be used to achieve lower error rates with shorter flows than passive schemes. Second, when multiple observation points are needed to detect relayed flows, passive traffic analysis schemes require continuous communication between these points to transmit flow statistics ($O(n)$ in the number of flows). Further, the detector must match each candidate flow against all of the candidates that must be correlated, requiring $O(n^2)$ computation. A blind watermark, however, requires no communication other than the shared key (i.e., $O(1)$) and can detect watermarks using $O(n)$ computation, since each flow is processed individually. This makes blind watermarks useful for large-scale traffic analysis.

2.1 Applications

Two particular applications of watermarks are stepping stone detection and linking of anonymized flows. In the former case, a watermark is inserted by border routers of an organization onto incoming flows, and checked for in outgoing flows (see Figure 2). The low false-positive rate provided by watermarks is paramount for large installations, since the volume of flows can make even a 10^{-3} false positive rate difficult to deal with. Scalability features are also important to large installations, especially ones with multiple border routers, since watermarks obviate the need for separate high-bandwidth channels between them.

With anonymous communication, the approach is similar: a malicious website or router can insert a watermark on a flow entering the network, and a cooperating router or observer can search for the watermark in outgoing flows, in an attempt to link the two. Modern anonymizing networks, such as Tor [5], require a large number of compromised routers in order to effectively attack anonymity, since for a fraction of f compromised routers, only about f^2 tunnels can be linked. Thus, an adversary would need to insert several high-bandwidth nodes or make use of a botnet [12]; in both cases, a scalable approach is needed. Furthermore, low error rates can make the results of such traffic analysis more useful; for example, a false-positive rate of 10^{-6} should erase any reasonable doubt that the identification of a user was a mistake, should it be used in a legal proceeding.

2.2 Previous work

Although there have been numerous previous proposals for watermark designs, an important feature gap remains. Many past watermark designs used large delays to achieve robustness to network noise. Such large delays present a barrier to practical deployment in stepping stone detection, as watermarks are applied to all flows, including benign ones, thus significantly impacting usability. The delays also make it easy to detect the presence of the watermark even without the shared key; thus, anonymizing networks can refuse to forward watermarked traffic as a countermeasure to the attack.

Many recent watermark designs used an interval-based design [15,19,24], which made them susceptible to a multi-flow attack [10]. By aggregating several flows, the interval-based transformations, applied at the same position, become easy to spot, and an attacker can even remove the watermark at a low cost. Other approaches that use packet-based approach have other problems. Wang et al.'s IPD-based watermarks [21] require tight synchronization between the watermarker and detector that can be destroyed by packet losses,

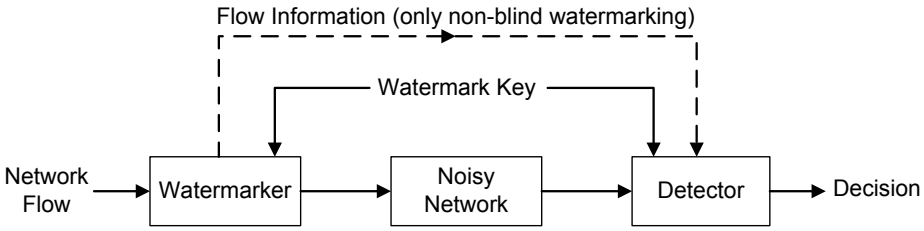


Figure 1. General model of network flow watermarking.

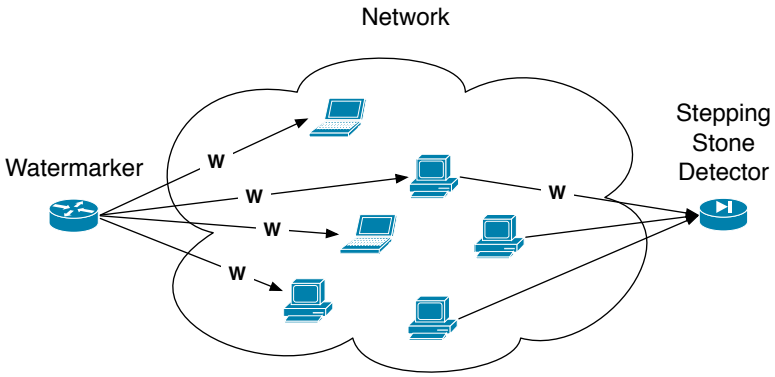


Figure 2. Stepping stone detection using flow watermarking.

reordering, etc. This watermark has also been shown to be susceptible to detection and removal [13]. RAINBOW [9] is a packet-based watermark that *is* robust to lost or reordered packets; however, RAINBOW takes a non-blind detection approach to achieve high detection accuracy over short flows while using very small delays. Our goal is to build a blind and therefore scalable watermark, at the cost of potentially requiring longer watermarked flows. Table 1 summarizes the properties of previous work.

3 SWIRL watermarking scheme

SWIRL is an interval-based watermark; therefore, it considers the flow as a collection of intervals of length T, with an initial offset o; i.e., the ith interval includes packets during time period $[o + iT, o + (i + 1)T)$. We first describe our approach to flow-dependent marking and then describe the overall SWIRL scheme.

3.1 Flow-dependent marking

To perform flow-dependent marking, we select two intervals: a *base* and a *mark* interval. The positions of these intervals will be fixed for all flows, but is otherwise arbitrary, with the restriction that the base interval must come earlier. During watermarking, we will use the base interval to decide which pattern to insert on the mark interval; the detector will correspondingly look for the pattern it computes using its version of the base interval.

The property of the base interval that we use is the interval *centroid*, which is the average distance of the packets from the start of the interval. I.e., if the interval i contains packets arriving at times $t_1, \ldots, t_n$, the centroid is:

$$C = \frac{1}{n} \sum_{j=1}^{n} (t_j - (o + iT)) \tag{1}$$

To decide on the pattern to be used on the mark interval, we quantize the centroid to a symbol s in the range $[0, m)$ for some $m \in \mathbb{Z}_+$. Since the range of the centroid is $[0, T)$, a simple approach would be to set $s = \lfloor mC/T \rfloor$. However, this would result in a non-uniform distribution for s, since a centroid is more likely to be in the center than at the interval. The actual distribution of centroids is heavily dependent on the rate of the flow as well as the distribution of packet delays. In order to approximate a uniform distribution for s, we take the approach of using finer partitioning. Namely, we set:

$$s = \lfloor qmC/T \rfloor \bmod m \tag{2}$$

Table 1. Watermark Scheme Comparison

Scheme	Invisible?	Robust to losses?	Resilient to MFA?	Scalable?
IPD-based watermark [21]	no	no	yes	yes
Interval-based watermarks [15, 19, 24]	no	yes	no	yes
RAINBOW [9]	yes	yes	yes	no
SWIRL	**yes**	**yes**	**yes**	**yes**

for $q > 1$. The quantization multiplier q helps smooth out the distribution; it is easy to see that as $q \to \infty$, s tends to a uniform distribution. We will discuss the choice of q in more detail in Section 5.1.

The value s is then used to transform the mark interval. We first subdivide the mark interval into r subintervals of length T/r each. The subintervals are then further subdivided into m slots each, with the slots numbered $0, \ldots, m-1$, see Figure 3. We select a slot in each subinterval i by applying a permutation $pi^{(i)}$ to s; each packet is then delayed such that it falls within a selected slot, possibly moving into the next subinterval. (Any packets at the end of the interval past the last selected slot are not delayed.) This produces a distinct pattern in the mark interval; see Figure 4 for an illustration. Note that we must use distinct permutation for each subinterval; otherwise we risk creating a periodic pattern that can easily be observed. The permutations $\pi^{(0)}, \ldots, \pi^{(r-1)}$ are part of the secret watermark key.

3.2 Detection

To detect the watermark presence, the detector analyzes packets in the base interval to compute the centroid $\hat{C}$. It then derives $\hat{s}$ from $\hat{C}$ using (2). It then counts the fraction of packets in the mark interval that are in the correct slot ($\pi^{(i)}(\hat{s})$). If this ratio, ρ, is greater than a packet threshold τ, then the watermark is considered to be detected successfully.

Note that the centroid of the interval may have shifted due to network noise. We therefore consider an alternate quantization of it, $\hat{s}'$, to be the next nearest quantization to $\hat{C}$:

$$\hat{s}' = \begin{cases} \lceil mq\hat{C}/T \rceil \bmod m & \text{if } \{mq\hat{C}/T\} \geq 0.5 \\ \lfloor mq\hat{C}/T \rfloor - 1 \bmod m & \text{otherwise} \end{cases} \quad (3)$$

where $\{x\} = x - \lfloor x \rfloor$ denotes the fractional part of x. We then repeat the detection using $\hat{s}'$ to compute ρ'. If $\rho' > \tau$, the watermark is also considered to be detected.

3.3 SWIRL design

A single watermark instance is likely to produce too high a rate of false errors. To improve detection, SWIRL uses n base and mark interval pairs. Let d be the number of intervals where the watermark was detected; then the entire watermark is considered to be detected if $d > \eta$ for some threshold η. The full list of parameters for SWIRL is summarized in Table 2. These parameters must be shared between the watermarker and detector. The system parameters are chosen to achieve a particular performance based on the properties of the original flows and the noisy channel. The secret parameters are chosen randomly and can be thought of as a secret key shared between the watermarker and the detector. In Section 6.1 we analyze the security of the SWIRL watermark by deriving the entropy of its watermark key.

Table 2. Watermark Parameters

System parameters	
r	Number of subintervals
m	Number of slots per subinterval
τ	Packet detection threshold
η	Mark detection threshold
n	Number of base and mark intervals
T	Interval length
q	Quantization multiplier
Secret Parameters	
o	Initial offset
b_j	Location of base intervals, $j = 0, \ldots, n-1$
m_j	Location of mark intervals, $j = 0, \ldots, n-1$
$\pi_j^{(i)} \in S_m$	Permutations for each mark subinterval, $j = 0, \ldots, n-1, i = 0, \ldots, r$

4 System analysis

4.1 False-positive errors

The false-positive error rate is the probability of detecting a non-watermarked flow as watermarked. To calculate this, first consider the probability that a single packet in some mark interval is in the "correct" slot. If we assume a Poisson distribution for the flows, it is easy to see that:

$$FP_p = \frac{1}{m} \quad (4)$$

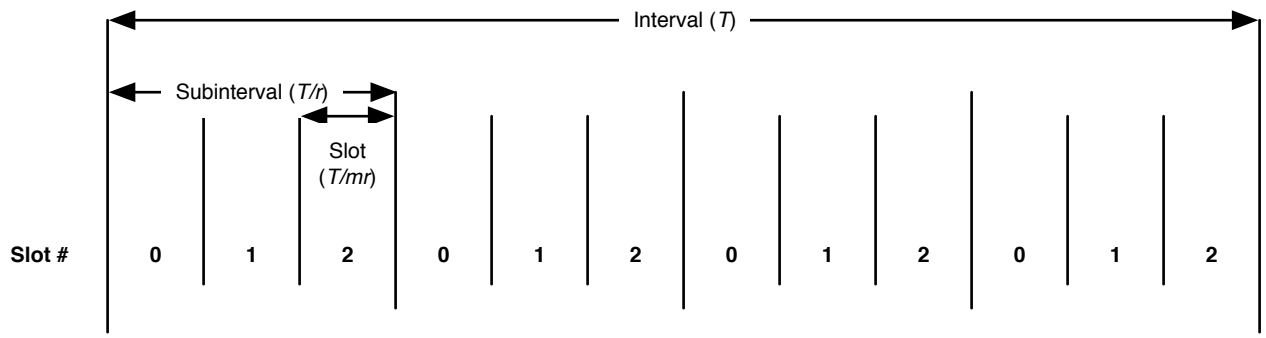
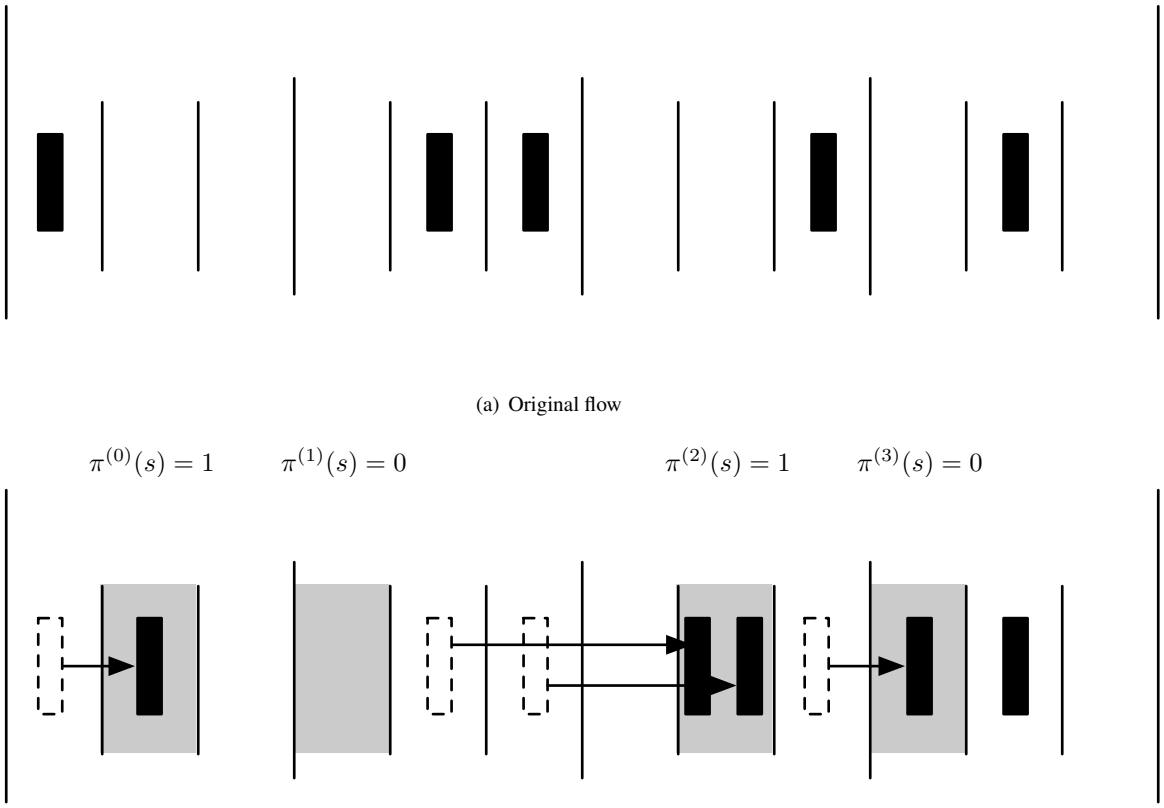


Figure 3. Slot numbering ($m = 3, r = 4$)



(a) Original flow

(b) Watermarked flow

Figure 4. Delaying packets to insert a watermark ($m = 3, r = 4$).

Of course, actual flows might have different distributions; however, unless the traffic patterns in a flow are correlated with the distances between slots (randomized by $\pi_j^{(i)}$), this will remain a good approximation.

Given a mark interval with P packets, the number of packets in "correct" slots will follow a Binomial distribution of P trials with probability of success PF_p, $B(P, FP_p)$. The cumulative distribution function of a Binomial distribution with v trials and success probability h, $B(v, h)$ is given by the regularized incomplete beta function $I(\cdot)$ as:

$$P(X \leq k) = I_{1-h}(v - k, k + 1) \tag{5}$$

It follows that the odds of getting at least τ fraction of pack-

ets in the correct slots can be computed as:

$$I_{1/m}(\lceil \tau P \rceil, 1 + P - \lceil \tau P \rceil) \tag{6}$$

Since we perform the detection for both $\hat{s}$ and $\hat{s}'$, the probability of an interval with P packets being considered detected is:

$$FP_I^P \leq 2I_{1/m}(\lceil \tau P \rceil, 1 + P - \lceil \tau P \rceil) \tag{7}$$

Modeling the flow as a Poisson process of rate λ, the number of packets in an interval of length T is distributed according to a Poisson distribution with parameter λT. Therefore, the overall probability of a false positive detection in an interval is:

$$FP_I = E_P^{\lambda T}[FP_I^P] = \sum_{P=1}^{\infty} \frac{e^{-\lambda T}(\lambda T)^P}{P!} FP_I^P \tag{8}$$

where $E_P^{\lambda T}$ computes the expected value with respect to P according to a Poisson distribution with parameter λT. Finally, the total number of detected intervals will once again follow a Binomial distribution $B(n, FP_I)$, thus the overall false positive rate is:

$$FP \leq I_{FP_I}(\eta, 1 + n - \eta) \tag{9}$$

4.2 False-negative errors

Now we consider the false-negative errors, i.e., the probability that a watermarked flow is considered not to be watermarked by the detector.

Again, we start by considering a single mark interval. The probability that it is considered not detected is:

$$FN_I \leq FN_s + (1 - FN_s)FN_p r \tag{10}$$

where FN_s represents the probability that neither $\hat{s}$ nor $\hat{s}'$ correspond to the original quantization s, and FN_p represents the probability that more than (1-τ) fraction of packets have shifted out of the "correct" slot, s.

Note that for the quantization to be misdetected, the centroid must shift by at least $T/(2mq)$; thus:

$$FN_s \leq P\left(\left|\hat{C} - C\right| > \frac{T}{2mq}\right) \tag{11}$$

Note that, given Q packets in the base interval, with delay of δ_j for packet j, we can calculate:

$$\hat{C} - C = \frac{1}{Q}\sum_{j=1}^{Q} \delta_j \tag{12}$$

We adopt a Gaussian approximation for the distribution of packet delays, as suggested in previous work [13]. Using synchronization described in Section 5.4, we can ensure that the distribution has mean 0. We thus model delay as i.i.d. Gaussian: $\delta_j \sim N(0, \sigma^2)$. Then:

$$\hat{C} - C \sim N(0, \sigma^2/Q) \tag{13}$$

$$\begin{aligned} FN_s &\leq P\left(|\hat{C} - C| > \frac{T}{2mq}\right) \\ &= E_Q^{\lambda T}\left[2\left(1 - \Phi_{0,1}\left(\frac{T\sqrt{Q}}{2mq \cdot \sigma}\right)\right)\right] \end{aligned} \tag{14}$$

where $\Phi_{0,1}(\cdot)$ is the CDF of $N(0,1)$, and $E_Q^{\lambda T}$ averages with respect to the Poisson distributed variable Q.

To compute FN_p, we first need to consider the probability that each individual packet would have shifted out of the assigned slot. Suppose that the packet p_j was distance x from the center of the slot ($-T/(2rm) \leq x \leq T/(2rm)$). Given the Gaussian distribution of δ_j, the probability of the shift is:

$$\begin{aligned} P(p_j \text{ shifted}|x) = 1 &- \Phi_{0,1}\left(\frac{(T/2rm) - x}{\sigma}\right) \\ &+ \Phi_{0,1}\left(-\frac{(T/2rm) + x}{\sigma}\right) \end{aligned} \tag{15}$$

Given that x will have a uniform distribution within the slot, we can integrate to find:

$$FN_{p_j} = \frac{rm}{T}\int_{x=-T/(2rm)}^{T/(2rm)} P(p_j \text{ shifted}|x)dx \tag{16}$$

The number of packets that are misdetected, out of P packets in the mark interval, is given by the Binomial distribution $B(P, FN_{p_j})$. Correspondingly:

$$FN_p = E_P^{\lambda T}\left[I_{FP_{p_j}}(1 + P - \lceil \tau P \rceil, \lceil \tau P \rceil)\right] \tag{17}$$

Using equations (14) and (17), we can compute FN_I in (10) and correspondingly:

$$FN = I_{FN_I}(n - \eta + 1, \eta) \tag{18}$$

5 Evaluation

We evaluate SWIRL watermarking scheme for the application of stepping stone detection. Our evaluation is also valid for Tor congestion attack prevention application, discussed in Section 7. For the application of linking flows in anonymous networks, a new set of parameters would need to be derived following the methodology described in this section.

5.1 Parameter choices

Table 3 shows the tradeoffs that result from choosing different parameters of the watermarking scheme, along with the chosen values for our implementation. The choice of q represents a tradeoff; on one hand, larger q increases the false negative rate by increasing FN_s of (14). On the other hand, smaller q may result in an uneven distribution of s, resulting in a multi-flow attack (MFA). We will defer a full examination of this tradeoff until our MFA analysis in Section 6.3; for the subsequent simulations and experiments, we set $q = 2.5$.

Likewise, r represents a tradeoff between false negatives and the amount of delay. The maximum inserted delay is bounded by $T/r(2 - 2/m)$. We experiment with different choices of r in the design.

In our experiments, we pick $n = 32$, i.e., 32 base and mark intervals are selected. This means that the watermark sequence must be at least $64T$ long; however, this ensures a low overall rate of errors.

The parameter T should be chosen based on the rate of the flow, since the false positive rate is proportional to $T\lambda$. In our experiments, we use flows that have a rate of 4–7 packets per second (pps), thus we set T to be 2s. For flows with rates lower than 3pps, we suggest doubling the T parameter to compensate for the smaller number of packets in each interval.

Both τ and η can be used to control the rates of false positive and false negative errors. For a fixed η, increasing the τ threshold improves the false positives while worsens the false negatives. Likewise, having the η threshold fixed increasing the τ threshold improves the false positives and worsens the false negatives. Figures 5(a) and 5(b) illustrate the effect on false errors that comes from varying each of the parameters while fixing the other; these results were obtained using a flow rate of 4.4pps (the average for the traces used in the following sections).

Note that given a choice of η, it is possible to find the value of τ that results in an equal rate of false-positive and false-negative errors; e.g., in Figure 5(a), this occurs at $\tau = 0.5$. The corresponding error rate is called the *cross-over error rate* (COER); in this case, it is approximately 10^{-7}. We can use this to optimize the joint choice of τ and η by computing the COER that can be achieved at any given choice of η. Figure 6 shows this for flows with average λ of 4.4pps. As can be seen, $\eta = 12$ minimizes the COER while the corresponding value of τ where the COER is achieved is approximately 0.5. Note that some applications will benefit from a different optimization target; e.g., lowest false-negative rate given a false-positive rate of at least 10^{-6}. In this case, the analytical false error rates can be used to find the optimal values of η and τ.

We also compute the η threshold that achieves the minimal COER for different flow rates. Figure 7(a) shows the results; the corresponding COER is shown in Figure 7(b).[2] This shows that, for optimal detection, η should be chosen based on the flow rate. The analytical computations, however, are based on approximate models of traffic and delays, and compute upper bounds of error rates. For simplicity of implementation, one might choose to use a single detection threshold regardless of the flow rate. Figure 8 shows the false positive and false negative error rates for a detector using a fixed $\eta = 12$, with the corresponding optimal η COER rates shown for comparison. As can be seen, for flows with smaller rates the fixed detection threshold improves the false negative errors rates at the price of increasing the false positive errors; this is opposite for the higher rate flows, but offers reasonable error performance overall.

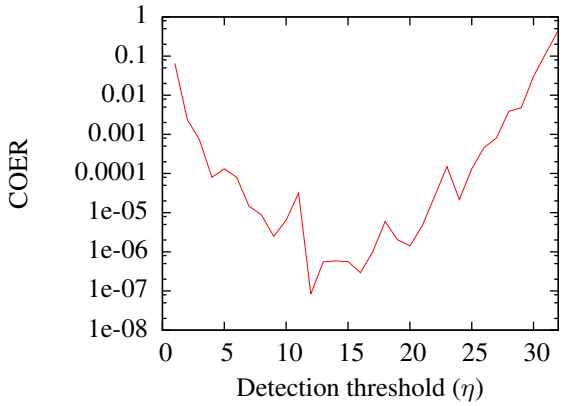


Figure 6. Cross-Over Error Rate (COER) for different detection thresholds η ($\lambda = 4.4pps$).

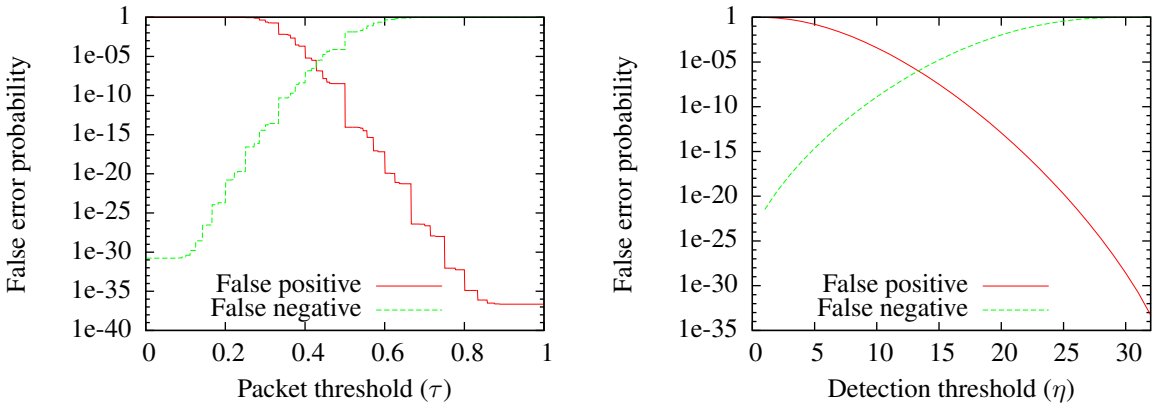
5.2 Simulations

We simulated the SWIRL watermarking system in Matlab. A watermark key is generated using the random number generators. We use $n = 32$, and use the system design parameters described in the previous section (see Table 3). We use traces collected by the CAIDA project from its `equinix-chicago` monitor—an OC192 link of a Tier 1 ISP—in January 2009 [18]. We selected SSH (port 22) flows out of the traces, since SSH is frequently used with interactive stepping stones; we used flows that were at least $2nT = 128$ s long, for a total of 304 flows. In every run of the simulation, an SSH flow is randomly selected from the database and is watermarked using the designated watermarking key. Since the analysis in Section 4 predicts that error rates are dependent on the rate of the flow,

[2] The non-monotonic behavior in the graph corresponds to changing the value of T for flows below 3pps.

Table 3. Tradeoffs in selecting watermark system parameters

Parameter	Tradeoffs		Selected value
	Increasing improves:	Decreasing improves:	
r	Delay, invisibility	False-negative errors	20
m	False positives	False negatives	5
τ	False positives	False negatives	0.5
η	False positives	False negatives	12
n	Detection performance (FP,FN)	Detection time	32
T	False-positive errors	Detection time	2 sec
q	MFA invisibility	False-negative errors	2.5



(a) $\eta = 16$ (b) $\tau = 0.5$

Figure 5. Analytical false error rates for different η and τ thresholds ($\lambda = 4.4pps$).

we chose flows that have similar rates for our simulations (9 pps $< \lambda <$ 10 pps).

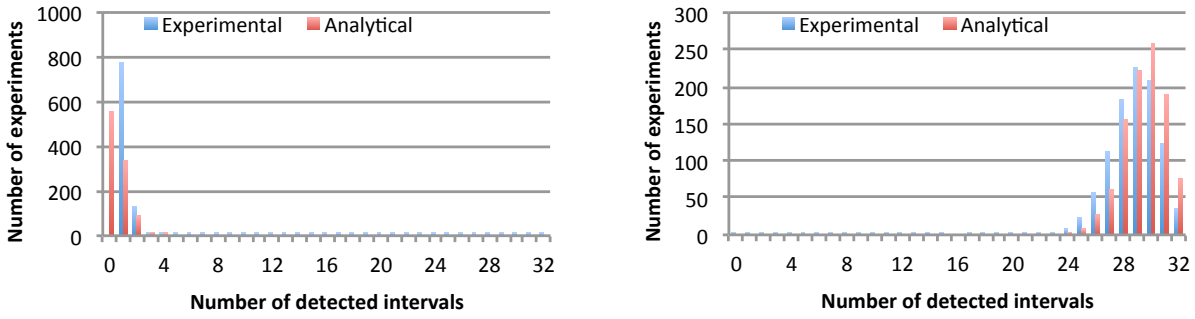
To simulate the effect of network delays, we captured traces of round-trip delays between pairs of randomly chosen PlanetLab nodes [2]; each trace captures the jitter properties of the relevant Internet path.[3] The traces have standard deviations ranging from $\sigma = 6.2$ ms to $\sigma = 12$ ms. For every run of the simulations a network delay sequence is selected at random and applied to the watermarked flow. Finally, the watermarked flow affected by network delay is evaluated by the simulated watermark detector to check for the shared watermark. We run this experiment 1000 times, each time with the same watermark key but random selection of network flows and network delays. Figure 9(b) shows the histogram of the number of watermark intervals (out of $n = 32$) that the detector successfully detects by evaluating a watermarked flow, namely *true detected intervals*. We compare this to the expected errors as predicted by the analysis in Section 4. The simulations show better than predicted error behavior due to the use of upper bounds in the analysis.

To consider false-positive errors, we perform the same simulations to evaluate the number of watermark intervals detected when the detector inspects non-watermarked flows. Similar to the previous experiment, we randomly select network flows from the database and apply a network delay trace to the selected flows using the same scenario. We then pass the flows through the watermark detector to check for the watermarked intervals. This experiment is also run for 1000 times. Figure 9(a) illustrates the experimental and analytical histograms of *false detected intervals*, namely, the number of watermark intervals detected by SWIRL detector from non-watermarked flows. Again, the simulations result in fewer errors than predicted by the upper bounds in the analysis. Comparing the two figures, it is easy to see that there is a strong separation between the two distributions; thus we should be able to achieve a low false error rate by choosing the detection threshold appropriately. Using a threshold of $\eta = 12$, we observed no false-positive or false-negative errors in our simulations.

5.3 Implementation

We implemented the SWIRL watermarking scheme over the PlanetLab infrastructure to evaluate its performance. We

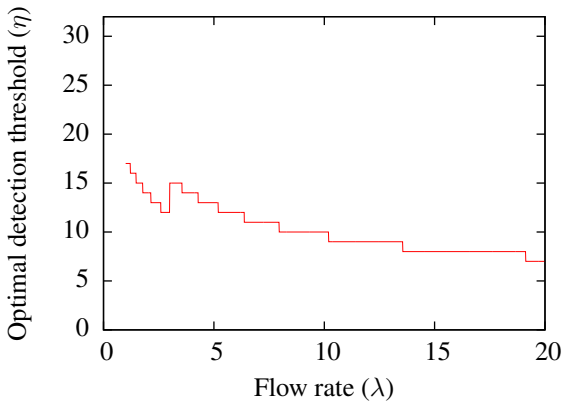
[3] We approximate the one-way jitter by the round-trip properties.



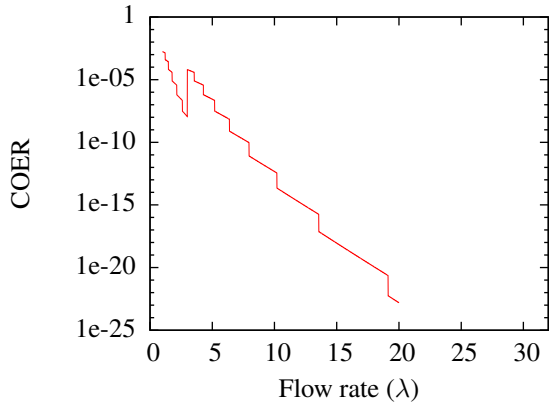
(a) False detected intervals

(b) True detected intervals

Figure 9. Histogram of watermark intervals detected by the simulated SWIRL detector for $\frac{T}{r} = 100$ (1000 random runs), as well as expected histogram values from the analysis.



(a) Optimal detection threshold



(b) COER

Figure 7. COER and optimal detection threshold for different effective rates.

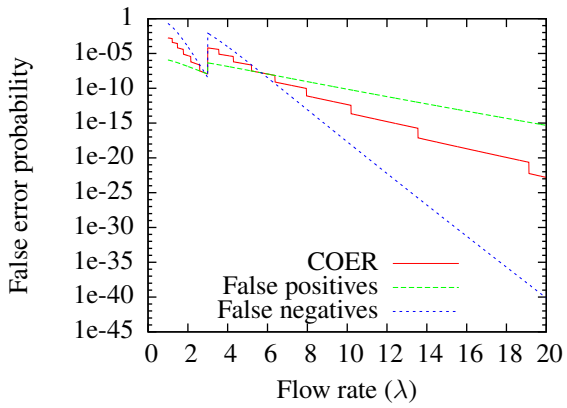


Figure 8. Probability of false positive and false negative errors for the optimal threshold (COER) and for a constant detection threshold of $\eta = 12$.

used same data set of SSH flows from the CAIDA traces, but we explored a wide range of flow rates. In each experiment, the watermarker reads the timings of packets in a flow read from the trace and then applies the watermark to them to generate a sequence of packets. These packets are then sent over the wide area to another PlanetLab node running the detector. Both the watermarker and detector are written in C++. As in simulations, we use the system parameters from Table 3. We also generate the watermarking key randomly as described before. To obtain false-positive rates, we fed flow timings from traces directly into the detector.

Table 4 summarizes the PlanetLab experiment results. Since the analysis suggests different detection performance for different flow rates, the flows are selected such that their

rates lie in one of the three ranges shown in Table 4; we used about 100 flows per group. Since the detection performance drastically improves with flow rate we skip data rates higher than 10 pps in our experiments. Also, in order to illustrate the effect of r parameter on the system performance each group of flows are watermarked with three different values of r. As the results show, in all cases a choice of $\eta = 12$ results in zero errors.

We notice that detection performance improves for higher rate flows, e.g., group A results in the best detection results. For a given group of flows, increasing r degrades detection performance, but improves watermark delay and invisibility as discussed in Section 6. Note that, as mentioned before, for the lower-rate flows, SWIRL detector uses a higher value for the T parameter, in order to compensate for the smaller number of packets in each interval. One can show that increasing T significantly improves the detection performance at the expense of longer watermark detection times.

5.4 Detector synchronization

As shown in Table 2, the offset o is shared between the watermarker and the detector. In fact, this is not necessary, as the scheme is self-synchronizing: the detector can perform detection using multiple offset value and return the best result. For example, Figure 10 shows a detector trying offset values in the range $[0, T]$ using steps of $T/100$. This approach allows the detector to use a randomized offset; this can serve as an additional countermeasure for the multi-flow attack, as discussed in [10]. It also ensures that two flows that exhibit similar behavior (e.g., repeated downloads of the same web page) will nevertheless be marked with different patterns.

6 Watermark invisibility

In this section, we start by showing that the very high entropy of the SWIRL watermark key makes is infeasible for an attacker to guess the watermark key. Then, we show that without having access to the watermark key an attacker is unable to detect the SWIRL watermark from a single watermark flow, as well as from multiple watermarked flows.

6.1 Watermark key entropy

To maintain invisibility, the watermark key must remain secret. We therefore estimate the size of the key space for the secret parameters used in SWIRL, as listed in Table 2.

First, we consider the space of permutations $\pi_j^{(i)}$. Each permutation is a random member of S_m, and each permutation is chosen independently. Therefore, the total space of permutations is $(m!)^{rn}$. Next, we must consider the space

```
1  intervals = range(2*n) # 0,...,2n-1
2  for i in range(n):
3      b[i] = intervals[0]
4      intervals.remove(b[i])
5      # pick m[i] uniformly at random
6      # out of remaining intervals
7      m[i] = random.choice(intervals)
8      intervals.remove(m[i])
```

Figure 11. Algorithm to generate interval assignments (shown in Python)

of parameters b_j and m_j. Note that it is possible to create equivalent keys by renumbering the intervals, therefore, we must count the number of non-equivalent interval assignments; we do so by defining a canonical ordering scheme such that $b_i < b_j$ for any $i < j$.

We can consider a recursive algorithm for generating a random assignment of $2n$ intervals into base–mark pairs, shown in Figure 11. It is easy to see that this algorithm generates every assignment with canonical ordering exactly once. The only random choice is on line 7 of the algorithm; at iteration $i (= 0, \ldots, n-1)$, there are $2(n-i)-1$ choices available. Therefore, the space of choices is:

$$(2n-1)(2n-3)\ldots(3)(1) = \frac{(2n)!}{2^n(n!)} \qquad (19)$$

For a conservative analysis, we can assume that $o = 0$ and that the first $2n$ intervals are chosen for watermarking; this results in the minimal required watermark duration of $2nT$ ($= 128$ s using the parameters in Table 3). We can thus estimate the entropy of the key choice as:

$$\log_2 \frac{(m!)^{rn}((2n)!)}{2(n!)} = rn\log_2(m!) + \log_2(2n!) - \log_2(n!) - n \qquad (20)$$

Using the parameters from Table 3, the key entropy is over 4000 bits, thus it is completely infeasible for an attacker to guess the secret key.

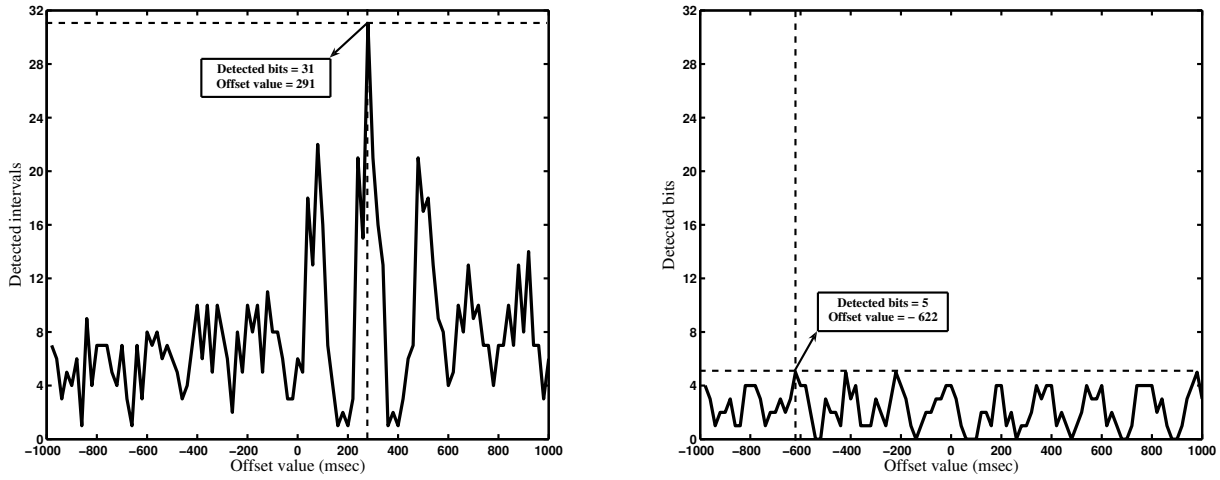
6.2 Single flow invisibility

In this section we demonstrate the infeasibility of distinguishing between SWIRL watermarked flows and benign flows by an attacker who does not have access to the watermark key.

Delay. As described in Section 2, a flow watermark is required to be invisible. The magnitude of delays makes a

Table 4. Watermark detection results for the PlanetLab experiments.

Group	Flow rate λ range (packet/sec)	r	T (sec)	True detected intervals		False detected intervals	
				Mean	Range	Mean	Range
A	6–10	10	2	31.6	30–32	3.5	0–6
		20	2	30.56	28–32	2.6	0–5
		30	2	31.4	29–32	2.8	2–4
B	3–6	10	2	30.89	29–32	2.87	1–5
		20	2	31.25	27–32	2.87	2–4
		30	2	29.4	24-32	2.87	1–4
C	0–3	10	4	25.25	15–31	1.4	0–2
		20	4	22.75	14–30	1.4	0–2
		30	4	20.66	14-27	1.1	0–3



(a) Watermarked flow

(b) Non-watermarked flow

Figure 10. Synchronization at watermark detection.

scheme easier or harder to identify. The maximum delay inserted by SWIRL is:

$$D_{max} = \frac{T}{r}\left(2 - \frac{2}{m}\right) \tag{21}$$

Table 5 shows the average watermark delay over the packets for different values of the redundancy parameter r, with T fixed at 2s. As evident from (17) and (14), lower r will reduce the number of false negatives at the cost of higher delay.

Information-theoretic tests. We also test the invisibility of the SWIRL using the information-theoretic tools designed by Gianvecchio et al. [8] for the detection of covert timing channels. We use two entropy tests of EN and CCE and apply them over a database of SSH flows, collected from real traces at the North Carolina State University (the average rate of the flows is $4.4pps$). The tests are evaluated for two classes of flows: a) regular non-watermarked flows, and, b) the same flows watermarked with SWIRL (each flow is 2000 packets long), with 10 tests per class. We then try different decision thresholds to decide whether a test metric corresponds to a watermarked flow. Figure 12 draws the ROC curves for the EN and CCE test metric, where the *true positive* is the odds of detecting a watermarked flow and the *false positive* is the odds of declaring a non-watermarked flow to be watermarked. As can be seen, the test metrics are not able to provide a confident separation between non-watermarked and SWIRL watermarked flows.

6.3 Multiple flow invisibility

Kiyavash et al. [10] show how multi-flow attacks (MFA) can be applied to compromise invisibility of interval-based flow watermarking schemes [15, 19, 24]. The main idea of the MFA attack is to collect a number of network flows wa-

Table 5. Average watermark delay (per packet) for different values of T/r along with the detection performance (σ = 10 msec, results averaged over 500 runs).

r	T/r (msec)	Average delay (msec)	Maximum delay (msec)	Mean true intervals (out of n=32)	Mean false intervals (out of n=32)
10	200	53.77	200	29.56	2.67
20	100	17.91	100	26.3	2.70
30	66.7	11.84	66.67	23.40	2.43
40	50	9.05	50	20.26	2.45

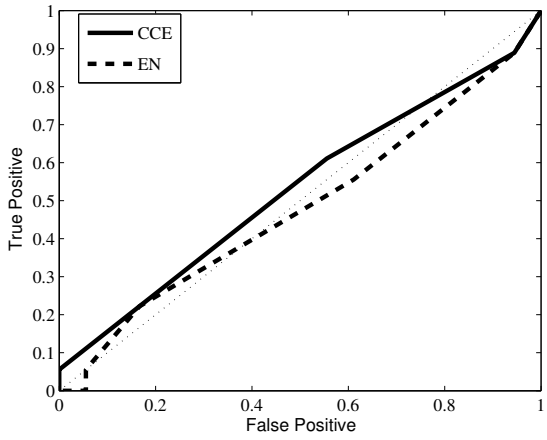


Figure 12. The ROC curves for the EN and CCE tests.

termarked by an interval-based watermarking scheme, using the same watermark key, and aggregate these flows to extract watermarking parameters and the watermark key. More specifically, the MFA attacker evaluates the *aggregate histogram* of a number of watermarked flows in order to find the watermark patterns, e.g., empty intervals, that are similar for all of the watermarked flows. The MFA attack has been shown to be highly effective in compromising previous interval-based watermarking schemes [15,19,24]. We analyze the resilience of SWIRL to multi-flow attacks.

The flow-dependent approach taken by SWIRL is designed to resist multi-flow attacks. In particular, since different flows have different watermarks, an aggregated histogram should not exhibit any repeated patterns. However, if the distribution of quantized values s is not uniform, an MFA attack may be able to identify the watermark. For example, if all watermarked flows use the same value of s for some interval, this will have a pronounced effect on the histogram.

The parameter q helps smooth out the distribution. In Figure 13, we plot the histogram of 10 non-watermarked flows, and 10 watermarked flows, using different choices for q. With $q = 5$, the variance of the watermarked flows is similar to the unwatermarked case. However, with $q = 1.25$, the histogram exhibits a clear pattern, since the number of quantification steps is too small and thus the distribution of s is heavily skewed.

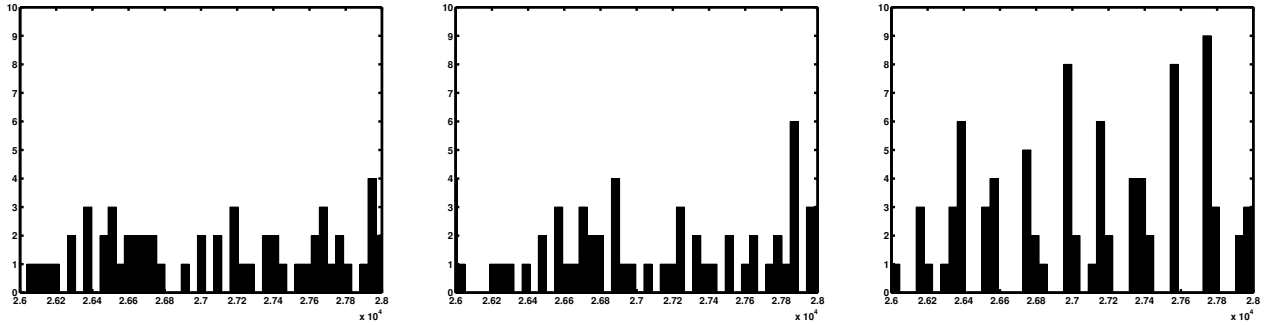
Table 6. Detected intervals for varying values of q ($\lambda = 4.1pps$, 1000 runs).

q	Watermarked		Non-watermarked	
	Mean	Range	Mean	Range
5	29.16	21–32	1.71	0–7
2.5	29.44	23–32	1.69	0–6
1.6	29.61	22–32	1.82	0–8
1.25	29.78	23–32	1.77	0–7

In this scenario, we watermarked all flows using the same offset. By choosing randomized offset, we can destroy the synchronization between flows: any shift of at least $T/(mr)$ will result in completely unaligned flows. As discussed by Kiyavash et al. [10], an adversary could still examine different potential alignments; however, when parameters from Table 3 are used, it would be necessary to examine 640^k alignments to find the correct alignment of k flows; this is both computationally expensive and is also deleterious to the false-positive detection rate for an MFA attack.

Note that, although increasing the q parameter improves resilience to the multi-flow attack, it also increases the false-negative rate. We demonstrate this in Table 6 by plotting the number of intervals counted as detected among a sample of both watermarked and non-watermarked flows. We note that the true detection rate falls with increasing q, whereas the false positives remain unaffected (for a given threshold η); this is consistent with the analysis in Section 4, where q factors into the false-negative but not the false-positive calculations.

Based on these results and the effect seen in Figure 13, we pick $q = 2.5$ to balance detection performance and susceptibility to the MFA attack.



(a) Non-watermarked histogram (b) Watermarked histogram ($q = 5$) (c) Watermarked histogram ($q = 1.25$)

Figure 13. Cumulative histogram of 10 flows, non-watermarked and watermarked with different values of q.

6.4 Active attacks

An adversary may use a more active approach to detecting and removing watermarks; e.g., by sending packets with embedded timestamps [13] to detect extra delays, or by introducing extra delays at the stepping stone. It is easy to see that, in the limit, the attacker can defeat any traffic analysis scheme by generating an independent packet schedule for a relayed flow, using dummy packets and introducing potentially large delays [3]. Previous work on stepping stone detection has considered limiting an attacker by a maximum tolerable delay [6]; however, we expect that a normal user would be less tolerant of added delays than a determined attacker, and a blind watermarking scheme that introduces delays that are much shorter than those it tolerates remains elusive and is an apt area for future research. We note that SWIRL will work well at detecting stepping stones and other relays over which the attacker does not have full control, as is the case in the application described in Section 7.

7 Tor congestion attack

Watermarks have been traditionally seen as privacy-invasive tools, since they can be used to link relayed flows and thus compromise anonymity systems such as Tor [5]. We show that SWIRL enables a new, privacy-enhancing use of watermarks in order to prevent a certain type of attack against Tor.

Evans et al. [7] demonstrated an attack on Tor that uses active probing to detect which Tor routers are used to forward a particular tunnel, thus breaking anonymity. Unlike watermarks or passive traffic analysis, their attack works even when the routers being used are *not* under the control or observation by the adversary. The basis of the attack comes from an earlier congestion attack, explored by Murdoch and Danezis [11]. However, a key feature of the new attack is the use of bandwidth amplification to create sufficient congestion as to make this attack practical on today's Tor network.

The bandwidth amplification exploits the fact that paths in Tor can be constructed to have an arbitrary length. This, coupled with the fact that each hop on a path knows only the previous and the next hop, makes it easy to construct a path that loops through a set of routers many times. This, in turn, ensures that a single packet sent by a user will result in k packet transmissions at each of the routers in the loop, for near-arbitrary values of k.

A potential defense described by Evans et al. is to modify the Tor protocol to restrict the number of circuit extensions it allows, and thus the maximum path length. However, they point out that this is not sufficient to completely prevent such congestion attacks, as loops can still be created by going outside the Tor network and then returning. In particular, a client can create a Tor tunnel, which forwards its traffic over Tor to TCP connection from an exit node to some destination on the Internet. This TCP connection can then be used to connect *back* to Tor as a client, and repeat the circuit again. Iterating this process yields the same functionality as the long-path attack. Although a naive approach may be foiled by exit and entrance policies in Tor, the attacker can instead use proxies, other anonymizers, or hidden Tor entry and exit points. Evans et al. leave defense to such external routing loops as an open problem.

We propose to use SWIRL as a solution. The basic strategy is to configure Tor exit nodes to insert a SWIRL watermark on all outgoing TCP traffic. Note that this labels the traffic as coming from Tor, but given that the list of exit nodes is published in the Tor directory, this does not significantly degrade privacy. Each entry guard, correspondingly, tries to detect the SWIRL watermark on an incoming TCP connection and rejects the stream if the watermark is found. This way, the congestion attack is restricted to in-

ternal paths only, which can be solved using the solution described above.

Note that this application can tolerate a significant rate of false positives (say, 10^{-3} or even higher). This is because a false positive will simply cause a legitimate user to retry a connection; given that the current Tor network does not provide very reliable service, an occasional extra failed connection is unlikely to significantly affect usability. This means that SWIRL parameters can be tuned to be able to mark shorter flows as compared with other setting. Additionally, full invisibility is not needed, as the open proxies are unlikely to be adversarial (if they were, they could simply generate the traffic themselves). Thus, the q parameter can be reduced to decrease false-negative errors.

It is important to realize that, although in principle any traffic analysis technique could be used, the properties of SWIRL make it particularly suitable for this task. Passive traffic analysis techniques (and non-blind watermarks) would require each exit node to communicate timing patterns of each exiting flow to each entry node. In addition to being very expensive, such an approach would completely defeat the protection provided by the Tor network, as each entry node would be able to detect which exit node each of its flows was using! A watermark, on the other hand, marks only the exiting flow and cannot be linked to the entry node. (Each exit node could, in fact, use the same watermark.) Additionally, other watermarking schemes introduce large delays, affecting the usability of the Tor network. The delays used by SWIRL, on the other hand, are significantly smaller than the typical latency of a Tor tunnel [22] and are unlikely to be noticed.

To study this attack, we simulated SWIRL being applied to Tor traffic flows. We used a set of flow timings observed by a Tor middle node[4] in our tests. We used a total of 14 flows that were long enough for our watermark. We then ran tests using both watermarked and non-watermarked versions of the flows to compute the number of true and false intervals detected. The results are shown in Table 7. The rates of the flows had a natural separation into two classes and we present the results for each class separately. Note that our tests are most representative for a direct connection from an exit to an entry node; any proxies or other relays may introduce extra delays that affect parameter choices. (However, the large and highly variable delays in the actual Tor network do not matter here, since the watermark is being transmitted over a channel external Tor.) Prior to implementation, it would be necessary to do a survey of proxy mechanisms available for congestion attacks and tune the parameter choices appropriately; we leave this to future work.

[4]This data set was provided to us by Steven Murdoch.

Table 7. Watermark detection results for Tor flows.

Flow rate (pps)	True intervals		False intervals	
	mean	range	mean	range
3.25–3.57	27.51	17–32	5.89	2–11
11.58-14.33	28.76	21–32	6.88	3–14

8 Conclusion

We proposed SWIRL, a novel flow-dependent watermarking scheme for network flows. SWIRL uses an interval-based structure in order to provide robustness to network perturbations, while evading multi-flow attacks by making the watermark dependent on the containing flow. SWIRL performs blind watermarking, reducing the communication overhead and computation overhead compared to passive traffic analysis or non-blind watermarking schemes. We show through analysis, simulation, and experiments that SWIRL is able to link related flows using flow lengths as short as 2 minutes, while providing error rates on the order of 10^{-6} or less. SWIRL introduces short delays on average and it is undetectable using existing covert channel detection tools. Finally, we show that SWIRL can be used to address a congestion attack on the Tor network.

Acknowledgments

We would like to thank anonymous reviewers for their comments on earlier drafts of this paper. We are also grateful to Nabil Schear for his help in processing the CAIDA data set and to Steven Murdoch for providing traffic data from the Tor network. This research was supported in part by National Science Foundation grant CNS 08–31488 and by the Boeing Trusted Software Center at the Information Trust Institute, University of Illinois.

References

[1] A. Back, U. Möller, and A. Stiglic. Traffic analysis attacks and trade-offs in anonymity providing systems. In I. S. Moskowitz, editor, *Information Hiding*, volume 2137 of *Lecture Notes in Computer Science*, pages 245–247. Springer, Apr. 2001.

[2] 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.

[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] 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.

[5] R. Dingledine, N. Mathewson, and P. F. Syverson. Tor: The second-generation onion router. In M. Blaze, editor, *USENIX Security Symposium*, pages 303–320, Aug. 2004.

[6] 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 17–35. Springer, Oct. 2002.

[7] N. Evans, R. Dingledine, and C. Grothoff. A practical congestion attack on Tor using long paths. In F. Monrose, editor, *USENIX Security Symposium*, pages 33–50. USENIX, Aug. 2009.

[8] 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, Oct. 2007.

[9] A. Houmansadr, N. Kiyavash, and N. Borisov. RAINBOW: A robust and invisible non-blind watermark for network flows. In G. Vigna, editor, *Network and Distributed System Security Symposium*. Internet Society, Feb. 2009.

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

[11] S. Murdoch and G. Danezis. Low-cost traffic analysis of tor. In V. Paxson and M. Waidner, editors, *IEEE Symposium on Security and Privacy*. IEEE Computer Society, May 2005.

[12] S. Murdoch and R. Watson. Metrics for security and performance in low-latency anonymity systems. In N. Borisov and I. Goldberg, editors, *Privacy Enhancing Technologies Symposium*, volume 5134 of *Lecture Notes in Computer Science*, pages 115–132. Springer, July 2008.

[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*. IEEE Computer Society, May 2007.

[15] 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. IEEE Communications Society, May 2007.

[16] J.-F. Raymond. Traffic analysis: protocols, attacks, design issues, and open problems. In H. Federrath, editor, *International Workshop on Designing Privacy Enhancing Technologies*, volume 2009 of *Lecture Notes on Computer Science*, pages 10–29. Springer, July 2000.

[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] C. Walsworth, E. Aben, kc claffy, and D. Andersen. The CAIDA anonymized 2009 Internet traces—January. `http://www.caida.org/data/passive/passive_2009_dataset.xml`, Mar. 2009.

[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. ACM, 2003.

[22] R. Wendolsky, D. Herrmann, and H. Federrath. Performance comparison of low-latency anonymisation services from a user perspective. In N. Borisov and P. Golle, editors, *Privacy Enhancing Technologies Symposium*, volume 4776 of *Lecture Notes in Computer Science*, pages 233–253. Springer, June 2007.

[23] K. Yoda and H. Etoh. Finding a connection chain for tracing intruders. In F. Cuppens, Y. Deswarte, D. Gollmann, and M. Waidner, editors, *European Symposium on Research in Computer Security*, volume 1895 of *Lecture Notes in Computer Science*, pages 191–205. Springer, Oct. 2000.

[24] 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.

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