New Attacks on Timing-based Network Flow Watermarks

Zi Lin
University of Minnesota
lin@cs.umn.edu

Nicholas Hopper
University of Minnesota
hopper@cs.umn.edu

Abstract

A network flow watermarking scheme attempts to manipulate the statistical properties of a flow of packets to insert a "mark" making it easier to detect the flow after passing through one or more relay hosts. Because an attacker that is willing to tolerate delay can (nearly) always eliminate such marks, recent schemes have concentrated on making the marks "invisible" so that a passive attacker cannot detect the presence of the mark. In this work, we argue that from a system's perspective, security against passive detection is insufficient for successful traffic analysis. We introduce a stronger, but feasible attack model (a *known/chosen flow* attacker) and a second security goal (security against *copy attacks*) and argue that security against both of these attacks is required for successful traffic analysis. We also demonstrate successful attacks against two recent watermarking schemes, RAINBOW and SWIRL, and show how considering these stronger attacks can aid in the design of passive detection attacks against each as well.

1 Introduction

Active traffic analysis, so called network flow watermarking, is the practice of manipulating the timing of a network flow so that the same flow, relayed by one or more intermediate hosts, can later be recognized. This technique has been the subject of increased interest in the past decade, because it requires low computational and communication cost while providing high accuracy in linking traffic flows. In these schemes, the packet timings of a network flow are modified, usually by buffering and delaying, to contain a distinctive pattern. If the pattern is later detected in another flow, we can conclude the two flows are the same with high probability. Flow watermarking is one of the most effective methods both for breaking anonymous communications systems [14, 15, 18, 7] and detecting network intruders launching a stepping stone attack [16, 12, 8, 7].

In contrast to the variety of schemes proposed, there is a relative lack of systematic study on attacking watermarking schemes. Since an active attacker can arbitrarily delay packets, thus destroying any watermark, recent schemes focus on how to evade passive detection and thus become "invisible" [15, 18, 7, 8]. However, many of these works consider a very limited attack model. Attackers are usually assumed to have access only to flows that comes from a black box in which a watermarker may have inserted marks to some of the flows. This assumption however, is often unrealistic: for example, in both cases mentioned above the adversary attacking the watermark has access to additional information about the marked flow. Furthermore, some adversaries (such as an anonymity network) may be better served by increasing the number of "unmarked" flows that appear to be marked, rather than trying to detect or remove a mark imperfectly.

Meanwhile, without a systematic view, it is often unfair to compare different detection attacks directly. In [11] Peng, Ning and Reeves studied how a stepping stone attacker, as a chosen flow attacker, could inject and analyze flows to detect the presence of a watermark and even replicate the parameters used by the watermarking system. Hereafter we refer to this attack as the PNR attack. In [9], Kiyavash et al. propose the *multi-flow attack* (MFA), which exposes a watermark by aligning multiple flows carrying the same watermark. A MFA can be launched by a single router in the network and is thus widely applicable. Very recently, Luo et al. [10], describe BACKLIT, a unique threat model in which the attacker acts as a known/chosen flow attacker and is thus able to detect state-of-the-art watermarking schemes. It is worth noting that the PNR attack model is stronger than BACKLIT, which is in turn stronger than the MFA model, so it is not surprising that their performance strengths also follow that order. On the other hand, MFA is more applicable than BACKLIT, which in turn is more applicable than PNR.

To address these concerns, we formalize two threat models for network flow watermarks. The first model, the *chosen flow attacker*, captures the capacities of a network intruder. This attacker may observe or even manipulate the input to the black box of a watermarker. In this case, packet delays due to deliberate watermarking and/or normal network processing become visible; and separating marked flows from unmarked flows becomes considerably easier. We apply this model to two recent watermarking schemes, RAINBOW [8] and SWIRL [7].

Both schemes use delaying as the basic operation to insert marks and are thus vulnerable to relatively straightforward chosen flow detection attacks.

Evaluating under the chosen flow attack may also help to develop ideas for new passive detection algorithms in the second threat model, the *isolated attacker*. For example, our initial chosen flow attacks on RAINBOW, which are based on testing jitter irregularity using cosine distance and histograms, led us to design a new passive detection attack based on testing the irregularities in the distribution of inter packet delays (IPDs). Similarly, a simple multi-flow chosen flow attack on SWIRL provides several important insights on a new multi-flow passive detection attack against SWIRL.

We implement and test our new attacks through experiments with real-world network traces. We show that chosen flow attacks can perfectly detect the watermarks with 100% recall and no false positives, outperforming the BACKLIT attacks. Our local, passive attacks on RAINBOW and SWIRL also achieve high ROC scores of ≥ 0.92; in some cases, ROC scores of 1.0 can be achieved.

Finally, we also introduce non-parametric *copy attacks*, which transfer marks between flows and eventually confuse traffic analysis without the knowledge of watermark secret keys and parameters. To our knowledge, this type of attack has not been studied before; yet every timing-based flow watermark is vulnerable due to the naïve attack that buffers enough traffic to mimic the inter-packet delays of marked flows. However, some schemes are vulnerable to less heavy-handed copy attacks. For example, the design of SWIRL allows us to demonstrate a very cost effective copy attack against it.

1.1 Paper outline

We briefly survey related work in section 2. In section 3, we establish the threat model for network flow watermarking schemes, identify new detection modes/attacks and a novel implementation of active copy attack. We describe our new detection attacks on RAINBOW and SWIRL, followed by evaluation on real-world datasets in section 4. We also present a detailed description of copy attacks on the aforementioned schemes along with experimental evaluations in section 5. Finally, we discuss possible defenses against these attacks and general defense strategies under our threat models in section 6.

2 Related Work

2.1 Stepping Stone Detection

Network intruders usually tunnel their attack traffic through one or more intermediate relays as "stepping stones", making the traffic origin hardly traceable. Within large enterprise networks, stepping stones are good candidates of compromised hosts. Network administrators therefore take stepping stone detection as part of their security monitoring routines.

Detecting stepping stone is usually done by linking outgoing flows with incoming ones. Often the intruder encrypted their tunnel (for instance by SSH). As a result, only packet counts, sizes and timings are available for flow characterization. By passive recording of these flow features, many schemes have studied the problem of linking streams [19, 4, 17, 4, 2, 6]. Since flow characteristics could be affected by padding schemes, packet retransmission and repacketization, and network jitter, successful passive stepping stone detection needs large number of observations, which in turn incur large overhead in both storage and computation. To address these efficiency issues, active approach is proposed as watermarking [16]. We will next briefly review the literature of flow watermarking schemes.

2.2 Network Flow Watermarking Schemes

There are two entities involved in flow watermarking, the encoder and the decoder. Both are typically boundary routers and share some common states. The encoder embeds a timing watermark to each incoming flow by introducing timing distortion (usually through delaying specific packets). At the other end, the decoder examines each outgoing flow to see whether it displays the unique mark and thus identify a potential stepping stone.

Several packet delaying schemes [16, 14, 8] intend to embed message bits by introducing distinctive network jitter. Those bits can be easily picked up by the decoder but they look unintentional to other routers. In [16, 14], skewness in jitter distribution, caused by delaying selected individual packets, is directly manipulated/measured. In RAINBOW[8], jitter distortion is expressed as an artificial jitter sequence (up to a few thousands in length), which is orthogonal to natural observed jitter in the linear space. This unique jitter, compounded together with normal network noise, can still be recognized by the decoder, using the inner product.

Interval-based watermarking schemes [12, 15, 18, 7] divide a flow into a series of time intervals and embed bits by manipulating the packet timing characteristics within each interval. Such approaches focus on intervals rather than packets, and are thus generally more resilient to packet insertion, losses and repacketization.

2.3 Attacks on Watermarking

In 2006 Peng et al. presented the PNR attack for stepping stone attackers [11]. The attack is designed to re-

cover the secret keys and system parameters of the particular scheme by Wang et al. [16]. By sending packets with controlled timing, uncommon extra delays can be detected by a variety of statistical and data mining tools. They also designed a duplicate attack to confuse the decoder by raising the false positive rate. The duplicate attack mentioned in [11] is simply repeat the watermarking with the extracted watermark parameters and is essentially different from the non-parametric copy attack we discuss here.

The Multi-flow attack (MFA), a passive detection attack recently proposed by Kiyavash et al. [9], demonstrates that an alignment of multiple marked flows shows an unusual synchronized pattern of busy and idle periods. This pattern is strong enough to conclude the presence of a watermark. MFAs show the potential of passive detection attacks and have helped motivate the design of newer watermarking schemes like RAINBOW and SWIRL [8, 7], that claim to resist these attacks.

Most recently, Lou et al. [10] studied how to expose watermarks in BACKLIT, an attack scenario in which the watermark encoder manipulates return traffic from a server while the attacker acts as a traffic relay between the client and the server. The watermark attacker can probe irregularities by comparing "clean" forward flows and "marked" backward ones. By doing so, BACKLIT gains extra knowledge similar to chosen flow attacker and is able to expose RAINBOW, SWIRL and other timing-based watermarks. The success of BACKLIT, however, relies heavily on the specific threat model.

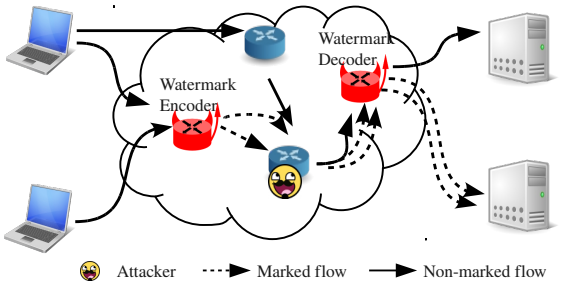

Figure 1: Isolated adversary: Accessible to output streams of the watermark encoder and input stream of the decoder.

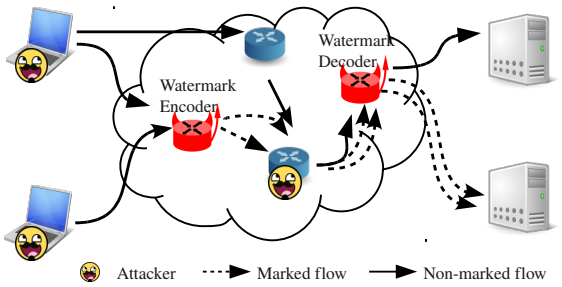

Figure 2: Chosen flow adversary: Inputs of the encoder are accessible too, in addition to outputs of the encoder and inputs of the decoder.

3 Threat Model of Network Flow Watermarking

In this section we first discuss the performance goals of network flow watermarking schemes. Then we briefly describe two threat models for watermarking schemes and define essential security properties that watermarks should achieve against adversaries. We note that anonymity systems and stepping stone intruders generally possess different capabilities and it is worthwhile to model them as different adversaries.

3.1 Performance Requirements

Active traffic analysis techniques, such as timing-based watermarking, aim to link network flows passing through one or more relay hosts efficiently and with high precision and robustness. This requires watermark detection to achieve both low false negative rates (FNR) and low false positive rates (FPR), even in the presence of network jitter and other distortion noise.

Low FNR requires the watermark not be easily erased by natural timing distortion. Detecting watermarked flows within a large network, although with keys, is sometimes challenging. Timing distortion such as delay by congestion, packet reordering, packet loss, and repacketizing are not uncommon. To achieve low FNR, usually multiple copies of a watermark are inserted into different flow locations, so that timing distortion will not affect the majority of the marks.

When facing a determined active adversary, FNR can be arbitrarily high because any timing-based mark can be erased through drastic measures. In an attempt to preemptively remove any timing-based watermark, active attackers can drastically change the timing by adding dummy packets, introducing large delays and/or sending packets in batches. Although effective, these active counter-measures against flow watermarking induce high costs that are typically unacceptable to the attacker, especially for delay-intolerant applications like Tor and SSH stepping-stones. Therefore, active attackers generally would prefer to appear passive and reactively launch counter-watermark attacks.

The requirement of low FNR against adversaries triggers the goal of passive invisibility: if watermarks ap-

pear invisible to passive attackers, active countermeasures may not be employed. Still, there exists a trade-off between low FNR and invisibility. The more watermarks that are embedded in the network flow, the more signals can be potentially picked up by the attacker.

On the other hand, low FPR is required so that non-watermarked flows are not frequently mistaken for marked ones; the advantage of watermarking over passive traffic analysis diminishes quickly as FPR grows.

Low FPR against adversarial manipulation, however, has not attracted much attention. Attackers that recovers specific watermark parameters can duplicate watermarks on different flows, "confusing the detector" [11, 9]. However, given the lack of knowledge of secret keys and parameters, blindly manipulating a benign flow into a watermarked flow is perceived to be hard for adversaries. We introduce a novel implementation of active duplication attack designed to increase FPR without the knowledge of any watermark parameter, called the *best-effort copy attack*. Busy Tor relays can use copy attacks to replicate and spread watermarks on all outgoing circuits and as a result, considerably increase FPR.

3.2 Types of Adversaries

The adversary against watermarking is assumed to control at least one host that relay the traffic between the encoder and decoder. Such assumption is realistic for both stepping stone intruders and anonymity system relays. The concrete threat model can be categorized into two classes: *Isolated adversary* illustrated in Fig. 1 and *chosen flow adversary* in Fig. 2. Adversaries in both threat models can be either pure observers (passive) or traffic manipulators (active).

Anonymity networks such as Tor are generally isolated active adversaries. Although Tor relays can manipulate packet timing, they seldom do so because their ultimate goal is to forward traffic as soon as possible. However that doesn't mean Tor cannot do anything actively about watermarking. As long as timing watermarks don't incur a high delay, a Tor relay could inject them and even further exchange watermarks between different circuits as we will show later.

The seemingly strong chosen flow adversaries are not uncommon. A stepping-stone intruder, for example, is capable of sending traffic at will. And it is usually true that those attackers get root privilege on "stepping stones" and are therefore able to observe and manipulate the packet timing. A careful stepping-stone intruder can set up trial connections to test the existence of a watermarker before actually using these "owned" workstations as stepping stones. The PNR attacker [11] is a good example.

3.3 Invisibility

In a nutshell, invisibility is defined as the ability to distinguish watermarked flows from non-watermarked ones. More formally, we define the Invisibility Game, played by an adversary (shown in Figure 3): Consider two sets of network flows S_0 and S_1; both of them are generated from the same distribution on flows. Flows in S_1 are manipulated by the watermarker while ones in S_0 are not. Both S_0 and S_1 are affected by similar network jitter. Now a random $i \in \{0,1\}$ is generated by a fair coin flip, the adversary is given one or more flows from S_i and she outputs i', she wins the game if $i = i'$. A watermark scheme is 'invisible' if no adversary can win the Invisibility Game with probability non-negligibly greater than "1/2". We further extend the definition of Invisibility Game to match up the ability of specific adversaries.

3.3.1 Invisibility with Isolated Adversaries

(Encoder) Output-only Detection To detect the presence of a watermark, there is little extra information the isolated adversary can obtain other than flow timing. Generally, we call such detection *Output-only detection* because the adversary only has access to (possible) outputs of the watermark encoder. Although such adversaries appear to be the weakest, their detection capabilities are still poorly understood.

Isolated passive invisibility has been the main focus in the literature to date. Various primitive analyses, such as entropy tests and distribution tests are utilized to examine the invisibility. These tests are usually carried out on individual flows. More powerful attacks come from the novel idea of collectively examining flows in S_1 or S_0.

Multi-flow Attacks. The Multi-Flow Attack (MFA) was introduced by Kiyavash, Houmansadr and Borisov [9] to show that previous interval-based watermarking schemes [12, 15, 18] lack invisibility when multiple network flows carrying the same watermark are carefully aligned. The aggregated histogram of packet frequencies will show repeated cleared/crowded intervals that can rarely happen without watermarking. The MFA shows the potential power of a passive isolated attacker.

We argue there is still room for many more intelligent detection attacks. In this work, we demonstrate some effective output-only detection attacks against state-of-the-art watermarking systems, RAINBOW and SWIRL, which have taken MFA resistance into account.

3.3.2 Invisibility with Chosen Flow Adversaries

Known Flow Attack. In this attack, chosen flow adversaries can choose to observe arbitrary flows; flows are

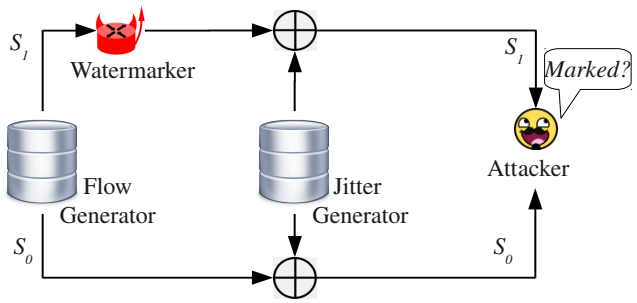

Figure 3: The Invisibility Game

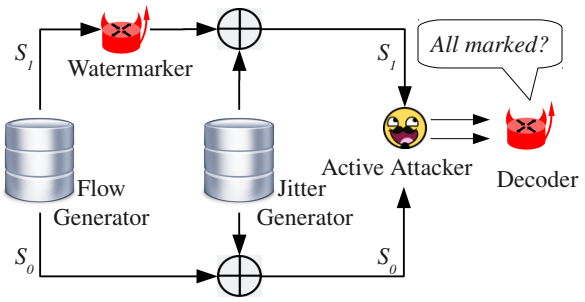

Figure 4: The Copy Attack Game

examined by pure observers before and after they are watermarked. We modify the invisibility game to reflect the fact that now adversaries also have access to flows before they pass through the watermarker. The adversary has the ability to compute the actual jitter imposed by the network and the watermarker. This brings opportunities to detect jitter-based watermarking schemes. In particular, we will show how RAINBOW, a jitter-based watermarking scheme, is vulnerable to such an adversary in section 4. We note that Houmansadr *et al.* [8] discuss global invisibility of RAINBOW in terms of the Kolmogorov-Smirnov test on inter-packet delays and jitter vectors. Unfortunately, K-S tests assume no *a priori* knowledge of the data distribution and therefore underestimate the power of adversaries against RAINBOW, limiting the usefulness of the result. In particular, we find discriminators that work almost perfectly to detect RAINBOW even when its parameters are carefully selected to avoid detection.

Chosen Flow Attack. In this scenario, adversaries can inject flows with a specific timing pattern and observe the distortion possibly added by the watermarker. The invisibility game with such an adversary gives the adversary the ability to intervene with flow generation. An example of a chosen flow attack is studied in [11]: when sending packets with known timing, extra delays caused by watermarks are distinguishable from normal network jitter. None of the known network flow watermarking schemes will resist this attack. For instance, we will show how SWIRL is visible under chosen flow attacks.

3.4 Active Copy Attack Resistance

As shown in Figure 4, the purpose of a copy attack is to confuse the decoder between flows from S_0 and ones from S_1. To our knowledge, copy attacks without knowing specific watermark parameters have not been studied for network flow watermark schemes. However, the concept of copy attacks on watermarking schemes is not new. Adelsbach et al. [1] introduced the notion of *protocol attacks* on multimedia watermarks where, for instance, an attacker can copy a watermark from one copy of a digital product to another, thus causing confusion and difficulties in the authorship/ownership dispute process. Peng et al. [11] designed a duplicate attack which can be seen as a copy attack by chose flow adversaries. Unlike the previous work, we show that it is possible to carry out copy attacks without knowing the secret key or concrete system parameters. naïvely, a straightforward replay of packet timing works for all timing-based watermarking schemes, including SWIRL and RAINBOW. However, we design a copy attack on SWIRL that is much simpler and extremely cost-effective. We show the effectiveness of this attack in section 5. With an active copy attack, two flows can replicate their watermarks to each other so the decoder is confused in flow linking.

4 Detection Attacks Based On Flow Characteristics

In this section we present detection attacks from different levels of adversaries. We demonstrate specific attacks on two exemplar schemes: RAINBOW and SWIRL. Following a brief recap of each scheme, we present a detection attack by a global adversary. We then show how the attack can be extended to work with a isolated adversary. All attacks are simulated and evaluated on CAIDA datasets [13].

4.1 Attack Implementation

We implement both watermarking schemes and attack algorithms in C++. To simulate the flow generator and jitter generator, we draw flows and jitter vectors from real-world network traces. Our simulation setup closely follows that of [7].

The network trace data are collected by the CAIDA project from its equinix-chicago OC192 link in January 2009 [13]. The dataset contains network flows that traverse in both direction of the link during a 4-hour pe-

riod. We selected SSH flows (destination port 22) because stepping-stone attacks are usually conducted over SSH, and got a total pool of 33 flows and 2.78 million packets. The packet rate ranges from 2 pps to 180 pps.

To simulate normal jitter caused by network delays, we adopt the RTT (round-trip time) measured between PlanetLab nodes [3] by Houmansadr and Borisov in their SWIRL work. In each simulation, a random trace of round-trip delays is chosen and applied to the watermarked flow.

4.2 RAINBOW

In the RAINBOW watermarking scheme [8], the watermark encoder and decoder share a database (DB) which records packet inter-arrival timing. In addition, they share a watermark key $w = [w_1, w_2, \cdots, w_n]$. The components of w take binary values of $+a$ and $-a$ uniformly:

$$w_i = \begin{cases} a & \text{with prob. 1/2;} \\ -a & \text{with prob. 1/2.} \end{cases}$$

For each individual incoming network flow, the encoder computes and stores the packet inter-arrival timing in DB, then inserts w as extra jitter. Considered the packet arrives at time $[t_1, t_2, \cdots, t_{n+1}]$, the inter-packet delays (IPDs) are $v = [t_2 - t_1, t_3 - t_2, \cdots, t_{n+1} - t_n]$. Now the flow is marked to carry the inter-arrival time as $v + w$, i.e. each inter arrival time is lengthened or shortened by a milliseconds. In case $v_i + w_i < 0$, w_i is ignored to avoid packet reordering, which would severely degrade the watermark if it happened too often. The details are shown in Algorithm 1.

On the other end, for each outgoing flow with inter arrival time vector v', the decoder computes a jitter vector $d = v' - v$ for each corresponding v recorded in the DB, and computes the cosine similarity score between d and w:

$$\cos(d, w) = \frac{\langle d, w \rangle}{\|d\| \|w\|} .$$

The detection algorithm is shown in Algorithm 2.

If v' is indeed the marked version of v, then we have $d = w + \delta$ where δ is the natural jitter introduced by the network delay. It is a widely adopted assumption that the distribution of jitter components can be modelled as a Laplacian distribution $Lap(0, \beta)$ [20, 8]. Subsequently we have

$$\cos(d, w) \sim Lap(\sqrt{a^2/(2\beta^2 + a^2)}, \frac{1}{\sqrt{2n}}) .$$

On the other hand, when the flow is not marked or an incorrect v is chosen, we have $d = v' - v + \delta$, and $\cos(d, w) \approx Lap(0, \frac{1}{\sqrt{2n}})$. The two distribution are nicely separated when the proper a value is chosen. The decoder decides the flow is marked if it scores higher than a threshold $\zeta = \frac{1}{2}\sqrt{a^2/(2\beta^2 + a^2)}$, and not marked otherwise.

4.2.1 Known Flow Attack

Under a known flow attack, the inter-packet delays before and after passing through the encoder are given, allowing the attacker to compute jitter. Therefore the task of detecting a RAINBOW watermark breaks down to distinguishing between vectors of the form $w + \delta$ and δ, where δ is normal jitter following a Laplacian distribution. In this case, there are several algorithms that could identify the watermark. We briefly introduce a detection algorithm that uses only one flow. Another detection algorithm that utilizes 2 or more flows and achieves perfect discrimination for most parameters is described in the appendix.

Single-flow Detection. We show how to detect the watermark with a single flow. Specifically, a histogram of jitter components within a time window serves as an excellent discriminator. In particular, we focus on the bin that counts the number of jitter components in the range $[-\beta/4, \beta/4]$.

By definition of the Laplacian distribution,

$$Pr(-\beta/4 < \delta_i < \beta/4) = 1 - e^{-1/4} \approx 0.221$$

so we expect over 1/5 components will fall in this bin. When the watermark w is added, each jitter component is translated to $\delta_i + a$ or $\delta_i - a$. Then under this new model, the probability that a watermarked jitter falls in the same bin is $Pr(-\beta/4 < x \pm a < \beta/4)$.

When we take $a >= \beta/4$, we have

$$\begin{aligned} & Pr(-\beta/4 < x \pm a < \beta/4) \\ = \; & Pr(a - \beta/4 < |x| < a + \beta/4)/2 \\ = \; & (e^{-a/\beta + 1/4} - e^{-a/\beta - 1/4})/2 \end{aligned}$$

For different a values, this probability can be directly estimated; see Table 1. The larger a is, the fewer jitter values will fall in the bin. Using observed jitter values, the drop in frequency is even larger, since originally the majority of the components fall in $[-\beta/4, \beta/4]$.

The attack algorithm is very simple: Scan through the packet arrival times with a moving time window and compute the percentage of jitter components in the range $[-\beta/4, \beta/4]$ within the window. If it is lower that a threshold θ, tag the packets within the window as "marked".

a =	0	$\beta/4$	$\beta/2$	β	2β
Frac.	0.221	0.197	0.153	0.093	0.034

Table 1: Expected fraction of watermark jitter in $[-\beta/4, \beta/4]$.

Algorithm 1 *RAINBOW-Embed* [a]

Input: v, w, n
  for $j = 1 \rightarrow n$ **do**
    if $v[j] + w[j] \geq 0$ **then**
      $v[j] = v[j] + w[j]$
    end if
  end for
  return v

[a] It is slightly different from the original algorithm described in [8]. However, this is the actual algorithm adapted in RAINBOW's source code and is in accordance with the experiment notes in [10] .

Algorithm 2 *RAINBOW-Detect*

Input: $v, DB = \{v_1, v_2, \cdots, v_m\}, w, \zeta$
  $isDetected = FALSE$
  for $i = 1 \rightarrow m$ **do**
    $d_i = v_i - v$
    $r = \cos(d_i, w)$
    if $r > \zeta$ **then**
      $isDetected = TRUE$
    end if
  end for
  return $isDetected$

Attack Evaluation. We simulated network flows by cutting randomly a subsequence of $n+1$ packets from a randomly drawn flow and we simulate the encoder with $n = 200$ to 1000 and $a = 2$ to 20. For each set of parameters, we simulated 1000 marked flows and 1000 unmarked ones. Also we simulated natural network delay on each flow, with the RTT dataset by Houmansadr and Borisov in their work of SWIRL [7]. The resulting flow is later ran against by the attack algorithm. The Laplacian model of jitter, estimated from the RTT dataset, indicates $\beta = 10ms$.

With a moving window that takes $m = 200$ consecutive packets, the attacker obtains a histogram of jitter values within the window and focuses on the number of small values within $([-2.5ms, 2.5ms])$, with $\beta = 10ms$. As indicated in Table 1, in the idealized situation we should see the percentage of such small jitter values drop from 20% to 9.3% or lower when an added watermark amplitude $a > \beta$ is expected. We set the threshold θ to be 10%. To illustrate the detection performance, we evaluate the single-flow detection attack by three criteria: True Positive Rate, False Positive Rate and Average Recall.

From Table 2, as expected the detection algorithm successfully recovers almost the entire watermark when $a \geq 10ms$, regardless of the length of the watermark. And it also does a very good job even when $a = 5ms$ and the watermark is sufficiently long and the performance drops when $a = 2ms$. It is evident that the known flow attack is effective against RAINBOW, without any knowledge of the watermark key.

a (ms)	length	TPR	FPR	Avg. Recall
2	200	0.004	0	0.001
	500	0.011	0	0.011
	1000	0.024	0	0.001
5	200	0.996	0	0.692
	500	1.000	0	0.846
	1000	1.000	0	0.921
10	200	1.000	0	0.984
	500	1.000	0	0.994
	1000	1.000	0	0.997
15	200	1.000	0	1.000
	500	1.000	0	1.000
	1000	1.000	0	1.000
20	200	1.000	0	1.000
	500	1.000	0	1.000
	1000	1.000	0	1.000

Table 2: Detection performance of single-flow chosen flow attack on RAINBOW

4.2.2 Output-only Detection Attack

Now we consider an attacker who can only see the flow after it passes the encoder: with only access to the potentially watermarked flow, we can still probe irregularities by sampling, with the assumption that the mark is not constantly present in the flow. In other words, we assume the flow contains marked segments and unmarked segments.

An important observation is made: The distribution of IPDs, $\{v_i\}$, is highly skewed. One example of the IPD distribution before and after RAINBOW watermarking is shown in Figure 5. In RAINBOW, a marked IPD v_i will be translated to $v_i + a$ or $v_i - a$. The skewness of these marked IPDs will be very different from the original.

Similar to the single flow detection algorithm, we utilize the change in histogram skewness as a predictor. Using a sliding window of w IPDs, histogram samples are generated. Viewed as vectors, these samples is naturally assumed to form a cluster in the linear space. Realizing the IPD distribution varies over time, we limit the histogram sampling to a flow segment of L packets. Since there is no single standardized model of IPD histogram we can refer to, we resort to using the centroid of the histogram samples to describe the cluster. The centroid is called the reference histogram, H_r. Further, each histogram sample is compared with the reference H_r. In case RAINBOW watermark is inserted, there exist samples that are much different from H_r. With a similar-

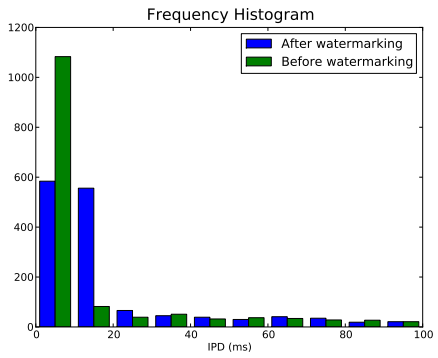

Figure 5: The histogram of first 2000 IPDs of one flow before and after watermarking (n=2000, a = 10ms). The bin size is 5ms.

ity function $\Psi(\cdot,\cdot)$ and a threshold τ_S, we judge a flow to be marked if there exists a histogram H_i such that $\Psi(H_i,H_r) \leq \tau_S$.

The watermarking effect on histograms can be viewed as a linear transformation on them. Suppose two histograms H and H' represent distributions of IPDs before and after the watermarking, respectively. Then there exists a matrix M such that $H' = H \cdot M$, leading us to use the cosine function as the similarity measure.

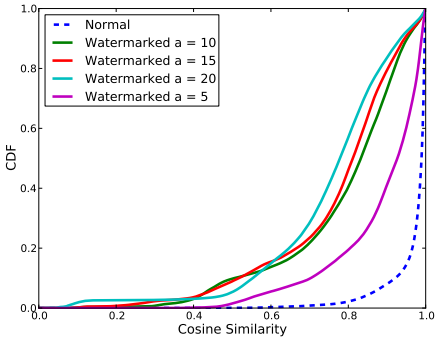

Figure 6: CDF with cosine similarity

A preliminary experiment confirms the effectiveness of cosine similarity on histograms. After selecting 100 random subsequences of $L = 4000$ packets from randomly drawn flows, we simulate RAINBOW watermarks (with $n = 1000$ and various a values) on each sequence. We take a sliding window of $w = 200$ IPDs. For the histogram sampling, we set the bin width to be 5ms, aiming to capture the watermark amplitudes $a > 5$ms. We also "clip" IPD histograms at $v_i \leq 200$ms as the majority of IPDs are in that range. The cumulative distribution functions of similarity scores over watermarked and non-marked regions respectively, are shown in Fig 6. The distance between the solid (watermarked) and dashed (unwatermarked) distributions suggests that we should be able to distinguish between the cases.

Evaluation of Output-only Detection Attack. We simulated 1000 flows by selecting a random subsequence of 4000 packets from a randomly drawn flow. We then simulate RAINBOW watermarks on the flow sample, with multiple parameter combinations. For each flow, normal network delay is simulated by imposing one sample jitter sequence from the RTT dataset.

We experimented with τ_S values between 0 and 1, and calculate ROC curves for each parameter combination. We selected two sets of them to show in Figure 7. To see how the outlier detection is affected by smaller segment, we also repeated the evaluation with 2000-packet flows and found similar results. The area under curve (AUC) values for various parameters for both experiments are summarized in Table 3. The performance of the attack agrees with the intuition: the chances of detecting watermarks improves with increasing amplitude and length. When $a = 20$ms, the AUC is generally close to or greater than 0.90. When $a \leq 5ms$, the performance of the attack drops substantially because the histogram with 5ms bin fails to reflect the relatively invisible distribution change.

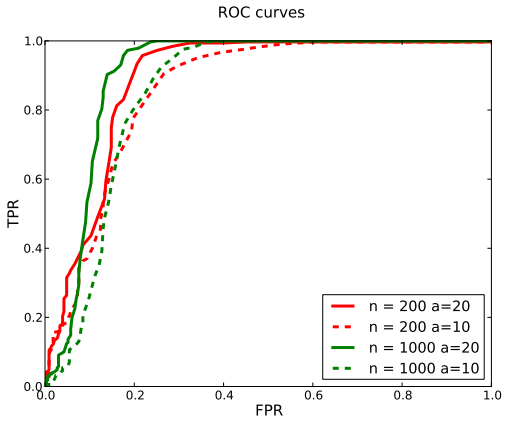

Figure 7: ROC curves of outlier detection

4.3 SWIRL

The design of SWIRL [7] can be briefly described as follows. Time is divided into *intervals* of two basic types: n "base intervals" in which seeds are generated, and n "mark intervals" in which packets are actually manipulated. Both being T seconds long, each base interval corresponds to one mark interval. Each mark interval is equally divided into r *sub-intervals*, and each sub-interval is equally divided into m *slots*. For each pair of base and mark intervals, r secret permutations, $\pi_1, \pi_2, \ldots, \pi_r$, are generated, such that each π_i is a permutation of numbers from 1 to m. Each base

a	length	AUC (L = 2000)	AUC (L = 4000)
5	200	0.417	0.432
	500	0.492	0.455
	1000	0.732	0.489
10	200	0.641	0.858
	500	0.711	0.878
	1000	0.728	0.895
15	200	0.894	0.889
	500	0.924	0.917
	1000	0.927	0.905
20	200	0.875	0.889
	500	0.913	0.917
	1000	0.920	0.910

Table 3: AUC scores of outlier detection

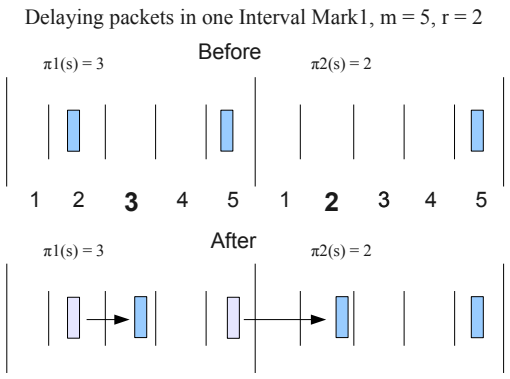

Figure 8: The marking procedure of SWIRL watermarking scheme

interval will be used to derive a seed $s \in Z_m$ and $\pi_i(s)$ will determine a *designated slot* for the $i-$th sub-interval of the corresponding mark interval. The encoder and decoder have agreed on the choices of n, T, m, r, and $\pi_1, \ldots, \pi_r$.

Within each mark interval, a watermark bit is inserted by re-arranging the packets to the designated slots for each sub-interval. More specifically, packets arriving before each designated slot are delayed into the subsequent one. The choice of seed s, the "quantized centroids" of base interval packets, is determined as follows: First, the centroid of packets, C, is computed as the mean arrival time of packets from the beginning of the interval. And the quantized centroid is

$$s = \lfloor qmC/T \rfloor \pmod m ,$$

where q is the quantization multiplier, introduced to increase randomness. The procedure of delaying packets into slots is illustrated in Figure 8.

For each outgoing flow at the other end, the decoder computes the centroids of every base interval and inspects the corresponding mark interval. A sub-interval is considered marked if the designated slot has packets. For each interval, a watermark bit is found if the number of marked sub-intervals is more than a pre-determined threshold τ. Finally, a watermark is detected if the number of watermark bits exceeds a preset threshold η.

4.3.1 Chosen Flow Attack

To make a SWIRL watermark stand out, the attacker can establish connections to a compromised host with uniformly paced traffic, sending K packets evenly per second. The interval centroid within such flows must lie within $[1/2, 1/2 + 1/K]$, regardless of the initial offset. As long as K is large, the quantized centroid will not change. So each marked flow will have the same pattern of cleared and occupied intervals. To see the pattern across different flows, we introduce the following attack that works like MFA based on packets, rather than time intervals, due to the fact that the pattern is introduced with a random offset on each flow.

1. One attacker injects multiple flows toward the other; the IPDs of all packets in all flows are t ms.

2. The receiving attacker models the jitter as Laplacian and estimates the variance β.

3. The receiver flags packets that arrive too soon ($\leq t - \delta$) or too late ($\geq t + \delta$ms) from the previous packet, with a threshold δ.

4. The receiver converts flagged packets into bit 1 and others into bit 0, transforming flows into bit strings. It examines the bit strings to see if there are common patterns of jitter.

Here is an example: We simulate an attacker that injects multiple flows with a rate of 20 packets per second (t = 50ms) to the SWIRL encoder and observes the outgoing traffic. We implement the SWIRL encoder with the recommended parameters ($n = 32$, $T = 2s$, $r = 20$ and $m = 5$). For each flow, the encoder applies a unique key (including new assignments of base/mark intervals) and a unique random initial offset $o \in [0, T]$. Network jitter is simulated by the observed jitter dataset. The chosen flow attacker first computes the jitter and estimate the jitter model: $Lap(0, 10)$ and set $\delta = 20$ms. Figure 9 shows three marked flows that are converted into bit sequences and put side by side for comparison.

The *a priori* likelihood of producing highly synchronized bit patterns by normal jitter is very small. The event of seeing a bit-1 (i.e. a jitter component $|j| > \delta$) happens with probability $p = e^{-\delta/\beta}$ by Laplacian distribution. Therefore each bit is the outcome of a Bernoulli

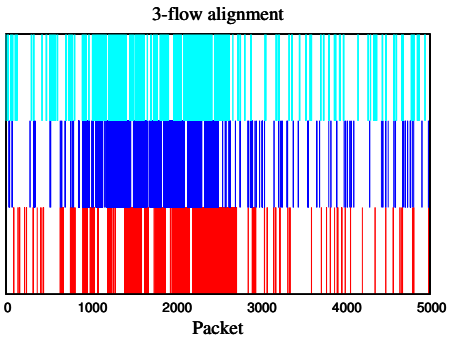

Figure 9: For each flow, an impulse is drawn when a packets is flagged. The synchronization of flags clearly shows the presence of watermark. On the contrast, the non-marked region on the right shows non-synchronization columns.

trial. With an alignment of k bit sequences, each column shows an outcome of k independent Bernoulli trials. The number of 1 bits in each column follows the binomial distribution $X \sim B(k,p)$. Subsequently, the probability of generating an all-1 column is $Pr(X=k) = p^k$. With observation of N columns, the expected total occurrence of all-1 columns is Np^k, with variance of $Np^k(1-p^k)$.

Returning to the example, we have $\beta = 10$, $\delta = 20ms$, $k = 3$, so $p = 0.135$ and the probability of generating one all-1 column is 0.00248. In Figure 9, the left 2500 columns of the alignment have 350 all-1 columns, much higher than the expected value 6.25 with a Z score of 140. In contrast, the right half has only 8 all-1 columns.

Learning from the example, the attacker uses a moving observation window and simply sets a Z score threshold. She raises the alarm when she sees the number of all-1 columns surpassing the threshold.

Evaluation of Chosen Flow Attack. We simulate the SWIRL watermarking on synthetic flows with two sets of SWIRL parameters: ($n = 32$, $T = 2s$, $r = 10$, $m = 5$) and ($n = 32$, $T = 2s$, $r = 20$, $m = 5$). Each synthetic flow lasts for 300 seconds and contains traffic with a uniform rate of 20 PPS. The resulting flow is further distorted by a random jitter vector from jitter dataset. In each experiment, 3 marked flows are converted to bit sequence as described previously and aligned. The attacker runs through the alignment with a moving window of size 1000. β is estimated as 10. We repeat the detection attack for three times on each alignment with $\delta = 10ms$, $20ms$, and $30ms$ respectively. The Z score threshold is set to 10. 3 non-marked flows with jitter are also fed to the attacker as control. We repeat the experiment 100 times.

There is no surprise to see that the attacker correctly identifies all watermarked flows and non-marked ones without any error, no matter which δ value is chosen. The accuracy of 100% is attributed partly to the fact that the Laplacian model usually over-estimates the tail distribution of natural jitter. That results in a much lower probability of seeing an all-1 column in the non-marked flow alignment, eliminating false positives.

4.3.2 Output-only Detection Analysis

Two important observations are made from the chosen flow attack on SWIRL:

- Because of the interval choice algorithm, mark intervals are likely allocated toward the end of the entire watermark period. Intuitively, out of 64 intervals, 1st interval must be base interval and the 64th must be mark interval. Very often, the last several intervals are all mark intervals.

- Initial offsets $\in [0,T]$ are not significant enough to break the synchronization of mark intervals.

Inspired by the observations, we examine the distribution of cleared and occupied sub-intervals and their relative positioning. With at least 75% probability, each designated slot is at least one half sub-interval length away from the neighboring designated slot. Intuitively we call this 'isolation', i.e. a packet in a marked region is either close to some neighboring packets because they are co-located in the same slot or far away from other packets. We slice the packets into groups by time, such that time gap between groups are at least $\mathscr{T}$ microseconds ($\mathscr{T}$ is chosen as a threshold.) Because of 'isolation', packets in marked intervals tend to form groups of short time-span.

To implement this heuristic, we use a naive clustering analysis algorithm that group packets by their arrival times. Two packets are grouped to the same cluster if their IPD in between is $\leq \mathscr{T}$. To capture the intuition of 'many short clusters', we use the maximum time-span among all clusters. If all clusters' time-spans are short, the maximum is guaranteed to be short. For one flow, we first divide it into one-second flow snapshots. Then we apply the analysis to each snapshot and assign the maximum cluster time-span to the snapshot as its heuristic feature. We denote this by F_i for the i-th snapshot. We hypothesize that F_i has a lower expected value if the i-th second is in a marked interval. To see this, we repeat the same procedure on k flows and compute $\bar{F}_i$ as the average F_i across multiple flows. The procedure is illustrated in Figure 10.

Here we give an example of such $\bar{F}_i$ vectors derived from 30 non-marked flows and 30 marked flows, as shown in Figure 11.

The output-only attacker collects marked flows passively and converts them to lists of heuristic values F_i with clustering parameter t. She computes the mean F_i

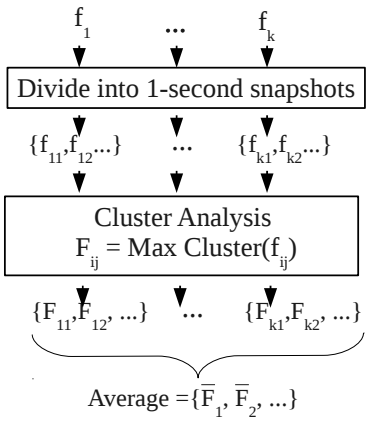

Figure 10: The procedure of computing $\bar{F}_i$

AUC Score				
t (ms)	k=10	k = 20	k = 30	k = 40
Sub-interval size = 100 ms (r=20)				
50	0.552	0.806	0.920	0.971
100	0.598	0.765	0.882	0.934
150	0.455	0.484	0.498	0.548
200	0.615	0.668	0.706	0.727
Sub-interval size = 200 ms (r=10)				
50	0.410	0.444	0.472	0.513
100	0.580	0.893	0.984	0.998
150	0.749	0.980	0.998	1.000
200	0.934	0.983	1.000	1.000

Table 4: AUC scores with combinations of t and k

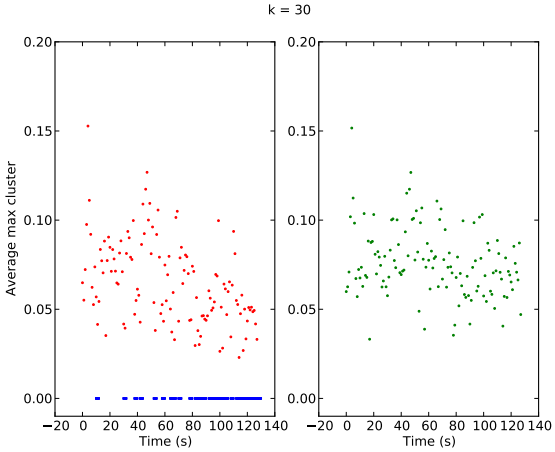

Figure 11: Two figures show $\{\bar{F}_i\}$ values derived from the same set of flows with and without watermark manipulation respectively. Marked flows has lower F_i values at marked intervals. (The thin dash line at y=0.0 level in the left figure indicates mark intervals.)

values across multiple flows. Marked regions are expected to score lower $\bar{F}_i$. Using a threshold τ_F, she determines that a flow is marked if there exists some $\bar{F}_i < \tau_F$.

Evaluation of Output-only Detection Attack. To test the output-only attack, we first generate flows with packet rate $\lambda \leq 10$ packets per second, the target packet rate for SWIRL. We then repeat for 100 times the simulation of SWIRL watermarking on those flows with the same two set of system parameters in Chosen Flow Attack evaluation.

Assuming no knowledge of the system parameters, the attacker launches the attack with various values of $\mathscr{T}$. Unfortunately we don't have an analytical model for setting τ_F so we use area under ROC curve (AUC) to evaluate the performance of the attack. The evaluation is shown in Table 4. We have encouraging results when the clustering threshold t is smaller than or equal to the subinterval length SI. Obviously, when $\mathscr{T}$ is larger than SI, the heuristic fails as the 'isolation' is bounded by SI and is gone when we look for larger gaps. For the second parameter set, the attack is able to score AUC greater than 0.98 with $t \geq 100$ms and $k \geq 30$. It is worth mentioning that when $\mathscr{T} = 50$ms, the attack cannot identify watermarked flows, due to the fact that the slot size is 40ms, very close to $\mathscr{T}$. In that case, given the low traffic rate, F_i with $\mathscr{T} = 50ms$ in the marked regions looks essentially the same as in non-marked regions, producing random numbers between 0 and 50ms. Finally, we are always able to find a threshold that allows the attack to separate marked flows and non-marked flows without error, when enough marked flows are accumulated ($k \geq 30$) and $\mathscr{T}$ is between the slot size and the sub-interval size, e.g. 100ms.

We further argue that blindly setting $\mathscr{T} = 100ms$ will be suitable to detect most SWIRL watermarks. First of all, heuristically we see the sub-interval should not be smaller than 100ms. The reason is that a smaller sub-interval is closely related to larger false negatives. Smaller sub-intervals result in smaller slots, making the scheme vulnerable to natural network jitter. Even worse, many smaller sub-intervals will be unmarkable because there is no packet falling into them, resulting in detection difficulties. On the other hand, we also reason the slot size should not be much larger than 100ms, since larger slots lead to much worse invisibility [7].

5 Active Copy and Ambiguity Attacks

Copy attacks are common in the area of media watermarking. However, to our knowledge the copy attack on network flow watermarks has not been studied extensively. In this section, we focus on copy attacks from a isolated adversary's perpective, resulting in stronger attacks (based on weaker assumptions). For convenience

of discussion, we refer to the original watermarked flow as the *Source Flow* and refer to the to-be-copied-to flow as the *Target Flow*.

Generic Copy Attack The first passive copy attack is the replay attack, which simply replicates the IPD sequence of the source flow with the target flow. In the RAINBOW scheme, the decoder automatically links incoming and out-going flows when a watermark is detected, since the detection only succeeds when outgoing timing and recorded incoming timing are matched correctly. In case of the replay attack, the decoder will have to face two identical watermarked flows and will no longer be able to do exact traffic linking.

Moreover, the replay attack doesn't need to be turned on all the time due to the fact that the watermark decoder will tolerate some errors. For example, suppose the entire watermark is imposed on the first 1000 packets of both source and target flows. When the attack changes the first 500 IPDs of the target flow with the source flow as a template, it creates a "chimera" flow that looks 50% similar to the source and the target, making the decoder more confused.

Copy Attacks against SWIRL. Unlike RAINBOW, SWIRL is interval-based, searching for intervals manipulated by the watermark encoding algorithm. SWIRL also tolerates errors caused by natural delay and jitters. Therefore it is possible to copy SWIRL watermarks imperfectly. Two key insights enable us to design a best-effort keyless copy attack efficiently:

- The interval centroid will not change even when a small fraction of packets are missing.
- The SWIRL decoder only watches for one packet to appear in the right slot and not all designated slots need to be filled.

In order to copy the mark, the attacker simply delays packets with a timing template recorded moments ago. The self-synchronization of the watermarking detection algorithm will automatically shift the interval boundary in place so we don't need to worry about the actual position of marks in the flow. The specific copy attack works as follows (also shown in Figure 12):

1. Between time u and $u+\Delta$, the attacker sets up a buffer that records the arrival times of a source flow as $Q = \{t_1, t_2, \ldots, t_n\}$ for packets coming within this period.
2. Between time $u+\Delta$ and $u+2\Delta$, each packet of the target flow arriving at time v_i will be examine to see if it can be matched to $t_j \in Q$ so that v_i can be delayed to make $t_j - u = v_i - u - \Delta$. Unmatched packets will be handled normally without extra delay. Meanwhile, we use another buffer Q' to record the timing of A in this period.
3. Replace Q with Q', u with $u+\Delta$, and repeat step2.

λ	Parameters
$3 < \lambda < 20$	$n = 32$, $T = 2s$, $r = 20$, $m = 5$
$20 < \lambda < 80$	$n = 32$, $T = 2s$, $r = 40$, $m = 5$
$80 < \lambda$	$n = 32$, $T = 2s$, $r = 80$, $m = 5$

Table 5: Adaptive setting of SWIRL system parameters

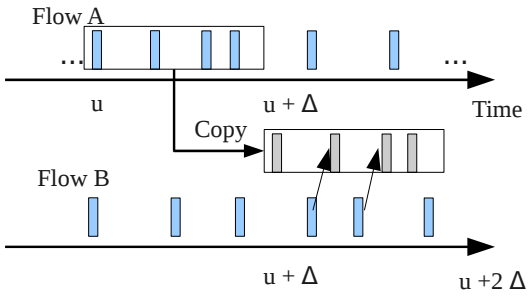

Figure 12: The *best-effort* copy attack on SWIRL scheme

This algorithm can run in real time and only requires two small buffers. Δ can be tuned to a small value, such as 500 milliseconds or 1 second. The "icing on the cake" is that the attacker can also copy the other way around simultaneously while copying from the source to the target.

Evaluation of Copy Attack on SWIRL. We implement the SWIRL encoder and decoder with multiple sets of parameters, adjusted to flows with different packet rates, shown in Table 5.

We simulated network flows by randomly choosing 128-second long intervals from randomly drawn flows. In total we obtained 300 flows for simulation and we simulated SWIRL watermarks on them, each with a unique watermark key. Our implementation of the copy attack chooses $\Delta = 500ms$. The attack is launched on every pair of flows 100 times. Eventually the average performance is calculated. The average number of bits detected on the target flow with the source flow's key is shown in Figure 13. Meanwhile, after the copy attack, the average bits detected by the target flow's key is also shown in Figure 14, organized by both flows' packet rates. The diagonal blocks show that it is easy to achieve copy effect between flows with similar system parameters. It also shows a clear trend that high-packet-rate flows ($\lambda > 40$) can easily copy any marks from a flow with a lower rate, achieving 25.4 bits detected on average. Also because of the high packet rate, they still remain watermarked by their original keys, with 19.5 bits. It is evident that the *best-effort* copy attack is agnostic to the actual settings

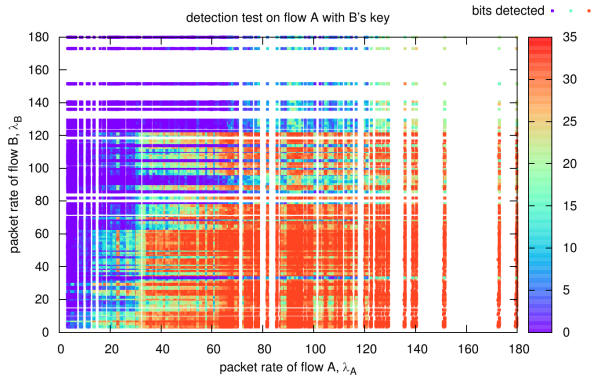

Figure 13: Bits detected after the target flow (A) copies

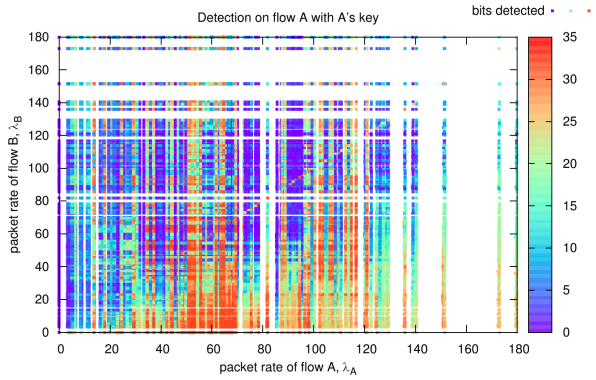

Figure 14: Bits detected after the best-effort copy attack, with the target flow's detection key.

of SWIRL such that two flows carrying completely different SWIRL watermarks can still exchange their marks without knowledge of SWIRL parameters.

6 Discussion

6.1 Defending Chosen Flow Attacks

Generally, defending against chosen flow attackers is hard given that timing manipulation is visible and measurable for such attackers. For jitter-based schemes, chosen flow attacker can directly compare the jitter in question with some theoretical model. One defense technique to borrow from stegnography is that the imposed artificial jitter should approximate the real jitter as much as possible. As to interval-based schemes, Kiyavash et al. in their work of MFA already suggested choosing different mark intervals for different flows [9]. However, both PNR attacks and our chosen flow detection algorithms indicate that using base/mark approach doesn't resist against chosen flow adversary. How to choose mark intervals randomly to get pass such adversary becomes an interesting open question.

6.2 Defenses against Isolated Passive Attacks

6.2.1 RAINBOW

There are two ways to mitigate our outlier detection attack on RAINBOW: one is to use a smaller amplitude value a, such as 5ms. The other is to only watermark a subset of IPDs. For example, only one in every k IPDs will be touched for watermarking. Either way we face a trade-off between invisibility and robustness.

6.2.2 SWIRL

To defend against the passive attack we described, an easy fix for SWIRL is to use larger offsets and more random interval assignments. Large offsets breaks the synchronization of marked intervals so multi-flow alignment is less effective. A large offset might, however, present a scalability issue for the self-synchronized watermark decoder because now trying all possible offset values will be a time consuming task. Another defense is to employ a more sophisticated mark interval selection algorithm. For example, we could choose 32 out of 100 intervals to be mark intervals, and have the choice of the 32 mark intervals be determined by the base intervals of individual flows. Properly evaluating these defense strategies will require further research.

6.3 Utilizing Copy Attack for Anonymity

Large autonomous systems (ASes) such as ISPs control the network routing infrastructure in a distributed manner. Studies have pointed out that AS-level adversaries could reduce the link anonymity of Tor substantially by passive traffic analysis [5]. Equipped with high-capacity routers, such adversaries can also employ watermarking to improve the linking results. Blind watermarking schemes such as SWIRL are more suitable than non-blind schemes like RAINBOW because decoders in blind schemes can act along with pre-configured parameters and keys, saving a huge amount of resources.

To defend anonymity against AS-level adversaries that use SWIRL, Tor relays can launch the copy attack against SWIRL to mingle the timing information from multiple concurrent flows and confuse the decoders. The impact of copy attacks against passive and active traffic analysis on low-delay anonymity networks like Tor is an interesting open problem, appealing for future investigation.

7 Conclusion

We have proposed a security evaluation framework for network flow watermarking schemes, based on threat modeling. We started from a strong adversary, who is

capable of *chosen flow attacks*, and proved the effectiveness of chosen flow attacks against the recent state-of-the-art watermarking schemes, RAINBOW and SWIRL. Using insights from these attacks, we were able to devise detection attacks from the perspective of weaker isolated passive adversaries that still defeat the invisibility claims of these schemes. Additionally, we point out the feasibility of keyless copy attacks against flow watermarking and the importance of defending against such attacks in a traffic analysis scenario. In particular we develop an efficient and simple copy attack that works very well against SWIRL. We were able to transfer watermarks from one marked flow to another non-marked flow. Such copy attacks are especially of interests to Tor relays, which have concerns about link privacy with potential watermarking from ASes or compromised relays. In our future work, we would like to investigate the attack effectiveness in the real world and evaluate the subsequent performance impacts.

8 Acknowledgements

We thank Nikita Borisov and Amir Houmansadr for kindly sharing the source code and datasets for RAINBOW and SWIRL. We thank our shepherd Nikita Borisov as well as several anonymous reviewers for their constructive feedback. This work was supported by NSF grant 0917154.

References

[1] A. Adelsbach, S. Katzenbeisser, and H. Veith. Watermarking schemes provably secure against copy and ambiguity attacks. In *Proceedings of the 3rd ACM workshop on Digital rights management*, DRM '03, pages 111–119, New York, NY, USA, 2003. ACM.

[2] 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.

[3] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman. Planetlab: an overlay testbed for broad-coverage services. *SIGCOMM Comput. Commun. Rev.*, 33:3–12, July 2003.

[4] D. L. Donoho, A. G. 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 *Proceedings of the 5th international conference on Recent advances in intrusion detection*, RAID'02, pages 17–35, Berlin, Heidelberg, 2002. Springer-Verlag.

[5] M. Edman and P. F. Syverson. AS-awareness in tor path selection. In E. Al-Shaer, S. Jha, and A. D. Keromytis, editors, *Proceedings of the 2009 ACM Conference on Computer and Communications Security, CCS 2009, Chicago, Illinois, USA, November 9-13, 2009*, pages 380–389. ACM, 2009.

[6] T. He and L. Tong. Detecting encrypted stepping-stone connections. *Trans. Sig. Proc.*, 55(5):1612–1623, May 2007.

[7] A. Houmansadr and N. Borisov. SWIRL: A scalable watermark to detect correlated network flows. In *Network and Distributed System Security Symposium*. Internet Society, Feb 2011.

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

[9] N. Kiyavash, A. Houmansadr, and N. Borisov. Multi-flow attacks against network flow watermarking schemes. In *Proceedings of the 17th conference on Security symposium*, pages 307–320, Berkeley, CA, USA, 2008. USENIX Association.

[10] X. Luo, P. Zhou, J. Zhang, R. Perdisci, W. Lee, and R. K. C. Chang. Exposing invisible timing-based traffic watermarks with backlit. In *Proceedings of 2011 Annual Computer Security Applications Conference (ACSAC'11), Orlando, FL, USA*, December 2011.

[11] P. Peng, P. Ning, and D. S. Reeves. On the secrecy of timing-based active watermarking trace-back techniques. In *Proceedings of the 2006 IEEE Symposium on Security and Privacy*, pages 334–349, Washington, DC, USA, 2006. IEEE Computer Society.

[12] 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.

[13] C. Walsworth, E. Aben, kc claffy, and D. Andersen. The caida anonymized 2009 internet tracesjanuary., Mar. 2009.

[14] X. Wang, S. Chen, and S. Jajodia. Tracking anonymous peer-to-peer voip calls on the internet. In *Proceedings of the 12th ACM conference on Computer and communications security*, CCS '05, pages 81–91, New York, NY, USA, 2005. ACM.

[15] X. Wang, S. Chen, and S. Jajodia. Network flow watermarking attack on low-latency anonymous communication systems. In *Proceedings of the 2007 IEEE Symposium on Security and Privacy*, SP '07, pages 116–130, Washington, DC, USA, 2007. IEEE Computer Society.

[16] X. Wang and D. S. Reeves. Robust correlation of encrypted attack traffic through stepping stones by manipulation of interpacket delays. In *Proceedings of the 10th ACM conference on Computer and communications security*, CCS '03, pages 20–29, New York, NY, USA, 2003. ACM.

[17] X. Wang, D. S. Reeves, and S. F. Wu. Inter-packet delay based correlation for tracing encrypted connections through stepping stones. In *Proceedings of the 7th European Symposium on Research in Computer Security*, ESORICS '02, pages 244–263, London, UK, UK, 2002. Springer-Verlag.

[18] W. Yu, X. Fu, S. Graham, D. Xuan, and W. Zhao. Dsss-based flow marking technique for invisible traceback. In *Proceedings of the 2007 IEEE Symposium on Security and Privacy*, SP '07, pages 18–32, Washington, DC, USA, 2007. IEEE Computer Society.

[19] Y. Zhang and V. Paxson. Detecting stepping stones. In *Proceedings of the 9th conference on USENIX Security Symposium - Volume 9*, SSYM'00, pages 13–13, Berkeley, CA, USA, 2000. USENIX Association.

[20] L. Zheng, L. Zhang, and D. Xu. Characteristics of network delay and delay jitter and its effect on voice over ip (voip). In *Communications, 2001. ICC 2001. IEEE International Conference on*, volume 1, pages 122 –126 vol.1, jun 2001.

A Appendix

Multi-flow Detection of RAINBOW. First, we consider an attacker that initiates two flows through the watermarker and receives them at the compromised host. Now he needs to determine if both flows carry the watermark by looking at jitter vectors d^0 and d^1. Similar to detection, we again use cosine similarity between d^0 and d^1 to test it. Therefore, we need to distinguish two hypotheses:

- H_0: d^0 and d^1 are unwatermarked flows.
- H_1: d^0 and d^1 are both watermarked flows carrying mark w.

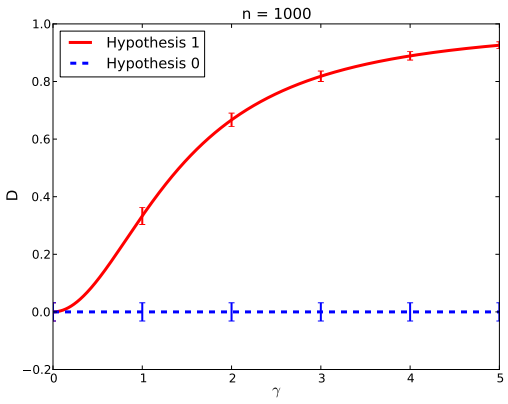

Figure 15: Cosine similarity between two jitter vectors

Define $D = cos(d^0, d^1)$, Under H_0, $d^i = \delta^i$ where δ^i are two independent jitter vectors. Under H_1, $d^i = w + \delta^i$. The decision rule is to use a threshold τ_D, such that if $D \geq \tau_D$, we reject H_0 and the watermark is present, and otherwise absent. To see this working we have the following lemmas (the proofs are shown in the appendix):

Lemma 1. *Suppose $X_1, X_2, \ldots X_n$ are i.i.d random variables with Laplacian distribution $Lap(0, \beta)$. Then $E(\sum_{j=1}^{n} X_j^2) = n \cdot (2\beta^2)$*

Proof.

$$\begin{aligned} E(\sum_{j=1}^{n} X_j^2) &= \sum_{j=1}^{n} E(X_j^2) \\ &= n \cdot (\sigma^2(X_j) + E(X_j)^2) \\ &= n \cdot (2\beta^2) \end{aligned} \quad (1)$$

□

Corollary 1. *For $\delta = [X_1, X_2, \ldots, X_n]$, $E(\|\delta\|) = \sqrt{n \cdot (2\beta^2)}$*

Lemma 2. *Suppose $X_1, X_2, \ldots X_n, Y_1, Y_2, \ldots, Y_n$ are i.i.d random variables with Laplacian distribution $Lap(0, \beta)$. For $Z = \sum_{i=1}^{n} X_i Y_i$, $\sigma(Z) = \sqrt{n} \cdot (2\beta^2)$*

Proof. Due to i.i.d random variables X_i, Y_i:

$$\begin{aligned} \sigma^2(Z) &= n\sigma^2(X_0 Y_0) \\ &= n(2\beta^2)^2 \end{aligned} \quad (2)$$

□

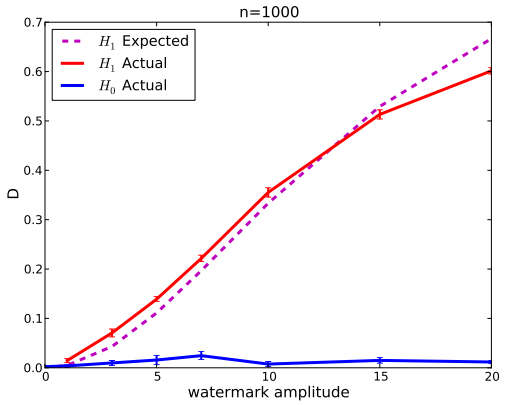

Figure 16: Statistics of RAINBOW multi-flow detection test

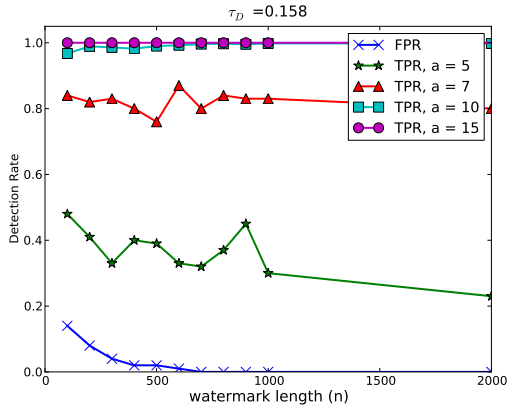

Figure 17: Performance of multi-flow detection attack

Lemma 3. *The inner product of jitter δ and watermark w, has distribution $\langle \delta, w \rangle \sim Lap(0, \sqrt{n}a\beta)$. The proof is referred to [8].*

Under H_0, the distribution of D satisfies the following characteristics:

$$\begin{aligned} E(D) &= \frac{E(\sum_{j=1}^{n} \delta_j^0 \delta_j^1)}{E(\|\delta^0\|)E(\|\delta^1\|)} \\ &= \sum_{j=1}^{n} E(\delta_j^0)E(\delta_j^1)/E(\|\delta^0\|)^2 \\ &= 0 \end{aligned} \tag{3}$$

$$\begin{aligned} \sigma(D) &\approx \sigma(\sum_{j=1}^{n} \delta_j^0 \delta_j^1)/E(\|\delta^0\|)^2 \\ &= \sqrt{n(2\beta^2)^2}/(\sqrt{n \cdot (2\beta^2)})^2 \\ &= 1/\sqrt{n} \end{aligned} \tag{4}$$

Under H_1, we have the following:

$$\begin{aligned} E(D) &= \frac{E(\langle w+\delta^0, w+\delta^1 \rangle)}{\|w+\delta^0\|^2 \cdot \|w+\delta^1\|^2} \\ &\approx \frac{na^2 + E(\langle \delta^0, \delta^1 \rangle)}{\sqrt{(\|w\|^2 + \|\delta^0\|^2)(\|w\|^2 + \|\delta^1\|^2)}} \\ &= na^2/(na^2 + n2\beta^2) \\ &= \gamma^2/(2+\gamma^2) \end{aligned} \tag{5}$$

where γ is the ratio of watermark amplitude to the Laplacian parameter β of jitter.

$$\begin{aligned} \sigma(D) &\approx \frac{\sqrt{2\sigma^2(\langle w, \delta \rangle) + \sigma^2(\langle \delta^0, \delta^1 \rangle)}}{(na^2 + n2\beta^2)} \\ &= \frac{\sqrt{4a^2\beta^2 + 4\beta^4}}{\sqrt{n}(a^2 + 2\beta^2)} \qquad (6) \\ &= \frac{2\sqrt{\gamma^2+1}}{\sqrt{n}(\gamma^2+2)} \qquad (7) \end{aligned}$$

Figure 15 shows how the two hypotheses lead to different D distributions under varying values of γ with $n = 1000$. Evidently H_1 is significantly different from H_0 when n and γ are sufficiently large. In that case, detection attack can be accomplished by a simple statistic test that reject H_0 with good confidence.

We set a threshold τ_D for dectection test. If two jitter vectors score higher than τ_D, the attacker rejects H_0 and deems them as "watermarked". Otherwise it accepts H_0. By Chebyshev's inequality, we expect FPR $\leq \frac{1}{2k^2}$ with $\tau_D = k/\sqrt{n}$. Note now we can choose the threshold value independent of actual a and β values used by RAINBOW.

Attack Evaluation. We simulated network flows by cutting randomly a subsequence of $n+1$ packets from a randomly drawn flow. For each set of n and a values, We simulated 1000 marked flows and 1000 unmarked ones, and ran the attack algorithm against them.

We first compute cosine similarity D between marked jitter vectors and between normal jitter vectors. Figure 16 shows that the actual detection performance fit nicely with the model. Therefore, we further set $\tau_D = 5/\sqrt{1000} \approx 0.158$, such that the false positive rate is $\leq 2\%$ when $n \geq 1000$. We then launch the detection attack with τ_D. The result, plotted in Figure 17, is satisfactory: Watermarks with $a \geq 10ms$ will be detected for sure while the false positive rate is nearly 0 when $n \geq 500$.