Multi-flow Attacks Against Network Flow Watermarking Schemes

Negar Kiyavash Amir Houmansadr Nikita Borisov
Dept. of Computer Science Dept. of Electrical and Computer Engineering
University of Illinois at Urbana–Champaign
Email: {kiyavash,ahouman2,nikita}@uiuc.edu

Abstract

We analyze several recent schemes for watermarking network flows based on splitting the flow into intervals. We show that this approach creates time dependent correlations that enable an attack that combines multiple watermarked flows. Such an attack can easily be mounted in nearly all applications of network flow watermarking, both in anonymous communication and stepping stone detection. The attack can be used to detect the presence of a watermark, recover the secret parameters, and remove the watermark from a flow. The attack can be effective even if different the watermarks in different flows carry different messages.

We analyze the efficacy of our attack using a probabilistic model and a Markov-modulated Poisson process (MMPP) model of interactive traffic. We also implement our attack and test it using both synthetic and real-world traces, showing that our attack is effective with as few as 10 watermarked flows. Finally, we propose a countermeasure that defeats the attack by using multiple watermark positions.

1 Introduction

Traffic analysis is the practice of inferring sensitive information from communication patterns. Traffic analysis has been particularly studied in the context of anonymous communication systems, where features such as packet timings, sizes, and counts can be used to link two flows and break anonymity guarantees [2, 22]. Traffic analysis is also sometimes used in intrusion detection, for example, to detect the presence of stepping stones within an enterprise [29].

Recently, there has been a growing interest in the use of *watermarking* to aid traffic analysis [27, 24, 21, 25, 28]. In this case, traffic patterns of one flow (usually packet timings) are actively modified to contain a special pattern. If the same pattern is later found on another flow, the two are considered linked. Watermarking significantly reduces the computation and communication costs of traffic analysis, and may also lead to more precise detection with fewer false positives.[1] Watermarking has been applied to both the problems of attacking anonymity systems [24, 25, 28] and detecting stepping stones [27, 21].

In both contexts, many flows must be watermarked before linked flows are discovered. In our work, we consider whether an attacker can learn enough information to defeat the watermark by observing multiple watermarked flows. (We use "attacker" here to refer to someone attacking the watermarking scheme; in the case where watermarks themselves are used by attackers, these will be the "counter-attackers.") We apply this multi-flow threat model to the latest generation of *interval-based watermarks* [21, 25, 28]. These watermarks subdivide the flow to be marked into discrete time intervals and perform transformative operations on an entire interval of packets. This approach is more robust to packet losses, insertions, and repacketization than previous approaches that focused on individual packets [27, 24], because the time intervals allow the watermarker and detector to retain synchronization. However, the same synchronization property can be used by attackers by "lining up" multiple watermarked flows and observing the transformations that were inserted.

We show through experiments that the interval-based watermark schemes are completely vulnerable to an attacker who can collect a small number of watermarked flows—about 10. This is sufficient to not only detect that a watermark is indeed present, but also to recover the secret parameters of the watermark scheme and to be able to remove the watermark at a low cost. Furthermore, our attack works even if different watermarked flows contain different embedded "messages," with only about twice the number of watermarked flows necessary.

We also consider some countermeasures to such attacks. We show that by using multiple "keys" (time interval assignments) to watermark different flows, it is possible to defeat our attack. This countermeasure comes at a cost of higher computation overhead at the detector and a higher rate of false positives. However, this increased cost is only linear, whereas the increased cost for the attacker is superexponential, thus providing an effective defense.

The rest of the paper is organized as follows. The next section presents the setting for our attack and reviews the three schemes considered in this paper. Section 3 describes the theoretical foundation for our attack, and Section 4 implements the attack. We discuss potential countermeasures to the attack in Section 5. Section 6 concludes.

2 Background

We first describe the setting of our attack in a bit more detail and then review the essential details of the watermarking schemes we analyze.

2.1 Network Flow Watermarking

The setting for network flow watermarking is similar to that of other digital media watermarks (and network flow watermarks use similar techniques). The general model, as shown in Figure 1, involves a network flow passing through a watermarking point (typically a router of some sort) that transforms, or *distorts*, the flow in some way (typically by modifying packet timings by selectively delaying some packets). In the general setting, the watermarker has a secret *key* and uses it to encode a *message* in the traffic characteristics.

After watermarking, the flow undergoes some natural or intentional distortion. Natural distortion can take the form of delays at intermediate routers (or rather, variability of delays, i.e., *jitter*), but may also include dropped or retransmitted packets, repacketization, and other changes. In addition, an attacker may intentionally distort traffic characteristics in order to prevent the watermark from being recovered.

The distorted flow finally arrives at a detection point. The detector shares the secret key and uses it to extract the message encoded in the watermark. A good watermark will allow reliable recovery of the message from the watermarked flow despite the intermediate distortion.

In network flow watermarks, the *message* component of the watermark may be used in two ways. First, all watermarked flows may be marked with a single message. In this case, the detector's main goal is to decide whether the watermark is present or not by checking whether the decoded message is the correct one. Alternately, different flows may have a different message embedded, so that when a watermarked flow is detected, it can be linked with a particular marked flow. This comes at a cost of less reliable detection, since the single-message context creates more opportunities to detect errors. Our attacks are designed to work in both single-message and multiple-message contexts.

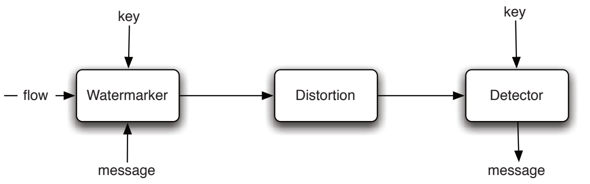


Figure 1: Network Flow Watermarking

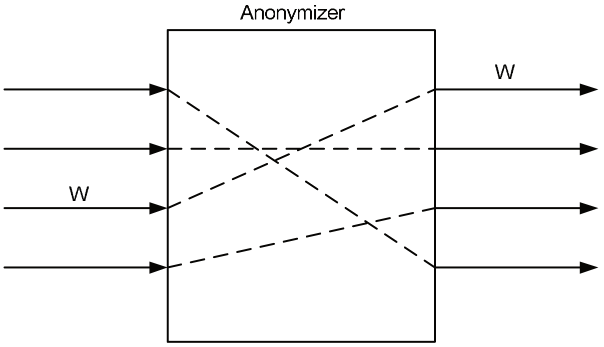


Figure 2: An anonymous system.

2.2 Watermarks in Anonymous Systems

At a very high level, an anonymous system maps a number of input flows to a number of output flows while hiding the relationship between them, as shown in Figure 2. The internal operation can be implemented by a mix network [8], onion routing [23], or a simple proxy [6]. The goal of an attacker, then, is to link an incoming flow to an outgoing flow (or vice versa).

A watermark can be used to defeat anonymity protection by marking certain input flows and watching for marks on the output flows. For example, a malicious website might insert a watermark on all flows from the site to the anonymizing system. A cooperating attacker who can eavesdrop on the link between a user and the anonymous system can then determine if the user is browsing the site or not. Similarly, a compromised entry router in Tor [11] can watermark all of its flows, and cooperating exit routers or websites can detect this watermark.

Note that this does not enable a fundamentally new attack on low-latency anonymous systems: it has been long known [23] that an attacker who can observe a flow at two points can determine if the flow is the same, unless cover traffic is used. (In fact, deployed low-latency systems such as Onion Routing [23], Freedom [1], and Tor [11] have all opted to forego cover traffic due to it being expensive, hoping instead that it will be difficult for an attacker to observe a significant fraction of incoming and outgoing flows.) However, watermarking makes the attack much more efficient. With passive traffic analysis, if one attacker observes n input flows and another observes m output flows, the attack will require $O(n)$ communication between the attackers and $O(nm)$ computation, as one attacker must transmit characteristics of all n flows to the other, and then each output flow must be matched against each input flow. With watermarking, on the other hand, no communication needs to take place between the two attackers after they have established a shared secret key, and the computation cost is $O(n)$ and $O(m)$ at the watermarker and detector respectively, as the watermarker marks each input flow and the detector checks each output flow for the presence of a mark.

Multi-Flow Attack In the above examples, a website or an input router will insert the watermark into all the input flows going through them. Therefore, it will be possible for the anonymous system to obtain multiple watermarked flows. These flows can then be used to recover the secret key and then remove the watermarks from subsequent flows, using the techniques we describe below. Our techniques are low-cost, requiring a small number of watermarked flows and modest computation, so it is easy to check whether watermarking is being applied by a given website or router by aggregating its flows.

The only context where our attack does not apply is in a *traffic confirmation attack*. In this case, an attacker already has a strong suspicion that a particular input flow corresponds to a particular output flow, and therefore need only watermark a single flow. Traffic confirmation attacks are a more rare use of traffic analysis, since they only confirm existing suspicions, rather than revealing new linkages between flows. Furthermore, the efficiency gains of watermarks are not beneficial in this case, since $n = m = 1$. Therefore, our attack will apply to the vast majority of practical uses of watermarks in anonymous systems.

2.3 Watermarks in Stepping Stones

A stepping stone is a host that is used to relay traffic through an enterprise network to another remote destination, in order to hide the true origin of the flow. To detect such hosts, an enterprise must be able to link an incoming flow to the relayed outgoing flow. The situation is therefore very similar to an anonymous communication system, with n flows entering the enterprise and m flows leaving. Once again, this task may be accomplished by passive traffic analysis [26, 29, 5, 12], but watermarks make such detection much more efficient. Passive techniques will require $O(nm)$ computation and potentially $O(n)$ communication, if there are multiple border routers through which traffic can enter or leave the enterprise. With watermarking, border routers for an enterprise will insert watermarks on all incoming flows, and check for the presence of the mark on all outgoing flows, as shown in Figure 3, reducing the computation cost to $O(n)$ and $O(m)$ for the incoming and outgoing flows.

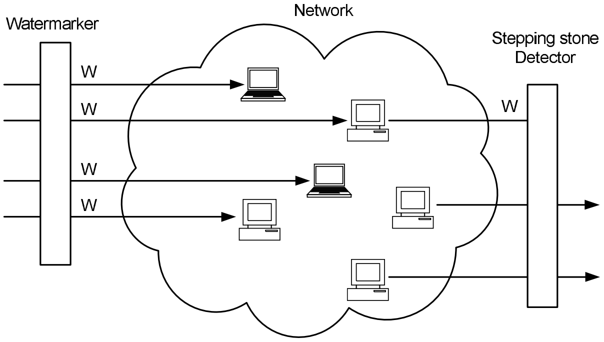


Figure 3: Stepping stone detection architecture.

Multi-Flow Attack Since all incoming flows must be marked, an attacker in control of a compromised host can simply generate multiple external flows destined for that host (and not relay them), and then collect the timing characteristics of the flows as they arrive at the host to recover the secret watermark key. Once this is accomplished, the key can be used to remove watermarks from relayed flows, thus defeating stepping stone detection.

2.4 Interval Centroid-Based Watermarking (ICBW)

We next review the scheme proposed by Wang et al. [25]; for more details of the scheme as well as some analysis we refer the reader to [25]. The scheme is based on dividing the stream into intervals of equal lengths, using two parameters: o, the offset of the first interval, and T, the length of each interval. A subset of $2n = 2rl$ of these intervals are chosen at random, and then randomly divided into two further subsets A and B each consisting of $n = rl$ intervals. Each of the sets A and B are randomly divided to l subsets denoted by $\{A_i\}_{i=1}^{l}$ and $\{B_i\}_{i=1}^{l}$, each consisting of r intervals. The i-th watermark bit is encoded using the sets $\{A_i, B_i\}$. Therefore, a watermark of length l can be embedded in the flow. Figure 4 depicts the random selection and grouping of time intervals within a flow for watermark insertion.

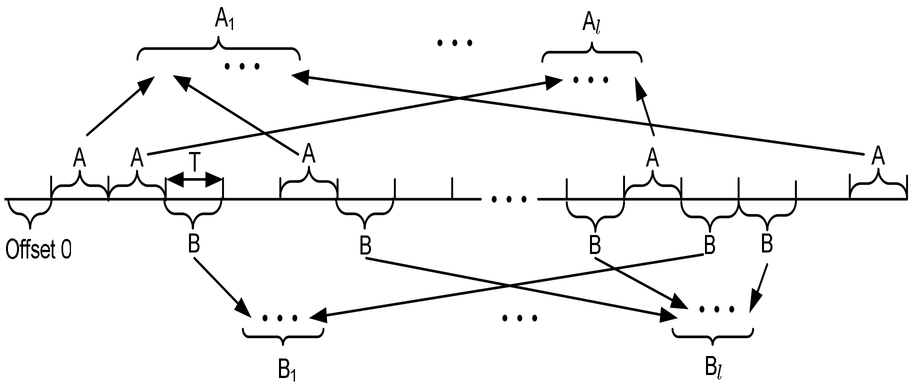


Figure 4: Random selection and assignment of time intervals within a packet flow for watermark insertion.

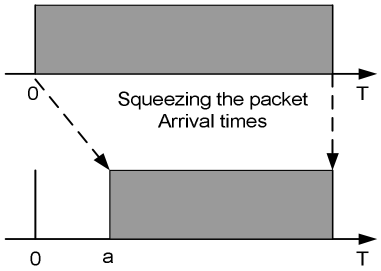
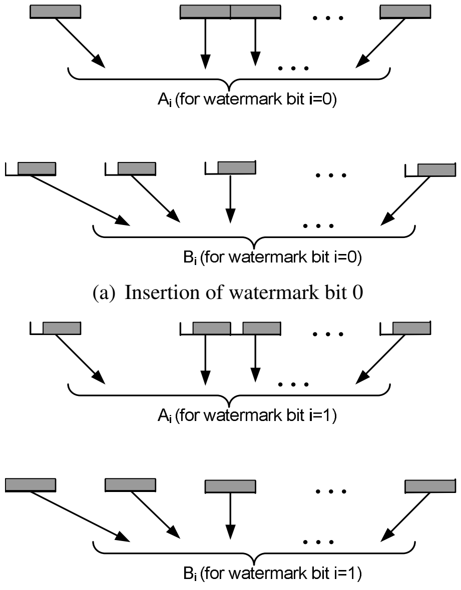


Figure 5: Distribution of packet arrival times in an interval of size T before and after being delayed.


Ai (for watermark bit i=0)
Bi (for watermark bit i=0)
Ai (for watermark bit i=1)
Bi (for watermark bit i=1)

(a) Insertion of watermark bit 0

(b) Insertion of watermark bit 1

Figure 6: ICBW bit insertion

The watermarker and detector agree on the parameters o, T and use a random number generator (RNG) and a seed s to randomly select and assign intervals for watermark insertion. To keep the watermark transparent, all of these parameters are kept secret. Depending on whether the i-th watermark bit is 1 or 0, the watermarker delays the arrival times of the packets at the interval positions in sets A_i or B_i respectively, by a maximum of a. Figure 5 illustrates the effect of this delaying strategy over the distribution of packet arrival times in an interval of size T (this operation is called "squeezing" by Wang et al.) Finally, the overall watermark embedding is illustrated in Figures 6 (a) and (b).

As the result of this embedding scheme, the expected value of aggregate centroid, i.e., the average offset of the packet arrival time from the beginning of the current length T interval, in either the intervals A_i (when watermark bit is 1) or B_i (when watermark bit is 0) corresponding to bit i is increased by $\frac{a}{2}$. The difference between the aggregate centroid of A_i and B_i now will be $\frac{a}{2}$ when watermark bit is 1 or $-\frac{a}{2}$ when watermark bit is 0.

The detector checks for the existence of the watermark bits. The check on watermark bit i is performed by testing whether the average difference of the aggregate centroid of packet arrival times in the intervals A_i and B_i is closer to $\frac{a}{2}$ or $-\frac{a}{2}$. If it is closer to $\frac{a}{2}$, then the watermark bit is decoded as 1 and if it is closer to $-\frac{a}{2}$, the bit is declared a 0. By focusing on the arrival times of many intervals (r of them for each bit of the watermark) rather than individual packet timings, the ICBW approach is robust to repacketization, insertion of chaff, and mixing of data flows. Network jitter can shift packets from one interval into another, but the suggested parameters for a and T (350ms and 500ms respectively) are large enough that few packets will be affected.

The secrecy of the interval positions A_i and B_i make the mark difficult to detect or remove, as it is hard to distinguish the patterns generated by the mark from natural variation in traffic rates. We show in Sections 3 and 4, however, that a simple technique allows an observer to effectively recover the watermark positions and values. This technique is applicable to any watermarking scheme that creates periods of clear or low traffic at *specific* parts of the flows across many flows. Next, we briefly describe *Interval-Based Watermarking (IBW)*, a flow watermarking scheme proposed by Pyun et al. [21] to detect *stepping stones*. Our attacks also applies to this scheme.

2.5 Interval-Based Watermarking

Similar to ICBW, the watermarking scheme of Pyun et al. [21] manipulates the arrival times of the packets over a set of preselected intervals. The watermark embedding is achieved by manipulating the rates of traffic in successive intervals. There are two manipulations: an interval I_i may be *cleared* by delaying all packets from interval I_i until interval I_{i+1}, or it may be *loaded* by delaying all packets from interval I_{i-1} until interval I_i. A loaded interval will therefore have twice the expected number of packets, and a cleared one will have none. To send a 0 bit in position i, the interval I_i is cleared and I_{i+1} is loaded; to send a 1, I_i is loaded and I_{i+1} is cleared. (Note that since clearing one interval implicitly loads the next, it takes 3 intervals to send a bit.)

The watermarker and detector agree on the parameters o, T and a list of positions $S = \{s_1, \ldots, s_n\}$; all of these parameters are secret. The watermarker encodes the watermark bits at the interval positions s_i and the detector checks for the existence of the watermark. The check is performed by testing whether the data rate in interval I_{s_i} differs from the rate in interval I_{s_i+1} by a factor exceeding a threshold; if it does, then a 0 or 1 bit is considered detected. By focusing on data rates rather than individual packet timings, the interval-based approach is robust to repacketization of data flows.

The detection process may generate false positives due to natural variation in packet rates, or false negatives, as delays between the watermarker and repacketization at the relay cause rates in intervals to shift. To ensure reliable transmission, each watermark bit is encoded in several positions in the stream. Pyun et al. show that this technique operates with very low false positive and false negative rates.

2.6 Spread-Spectrum Watermarking

In the DSSS watermarking technique due Yu et al. [28], a binary watermark is embedded in the flow to achieve *invisible* traceback. In their proposed approach, each bit of a length n binary watermark is embedded in an interval of length T_s. Hence the whole watermark is inserted in an interval of length nT_s. To embed a watermark bit 1, the rate of the packets in the designated interval of length T_s are manipulated according to a Pseudo-Noise (PN) code. The PN code is a quickly varying signal that switched between $+1$ and -1 and duration of each ± 1 period is T_c. In particular, Yu et al. [28] choose a length-7 PN code for their implementation. When PN code is $+1$, the rate of the flow remains intact, but when PN code is -1, the rate of the flow is decreased for a duration of T_c.[2] On the other hand, to embed a watermark bit 0, the flow is manipulated using the complement of the PN code. Figure 7 depicts the embedding of watermark 110 for a PN code of length 5.

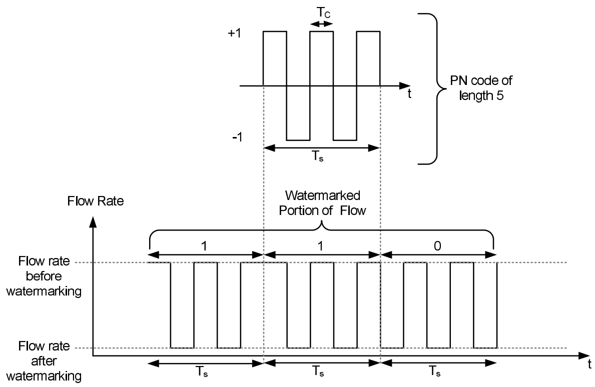

T_C
+1
-1
t
T_s
PN code of length 5
Flow Rate
Watermarked Portion of Flow
1
1
0
Flow rate before watermarking
Flow rate after watermarking
T_s
T_s
T_s
t

Figure 7: A length-5 PN code and insertion of DSSS watermark 110.

The watermarker and detector agree on the parameter T_s and a Pseudo-Noise code. The detector recovers the watermark by first applying a high-pass filter to the received signal and subsequently passing it through despreading and a low-pass filter. The details of the detector's structure are inconsequential to our attack and the interested reader is referred to [28].

Given that the watermark insertion technique in DSSS reduces the flow rates over certain intervals across all flows, it is vulnerable to our multi-flow attack.

3 Attack Analysis

In this section, we present a probabilistic analysis of our attack using a model for interactive traffic. Though some watermarked traffic may consist of non-interactive bulk transfer traffic, we will show in Section 4.1 that interactive traffic presents a more difficult case for our attack, and thus we analyze it here. As DSSS watermarks work well only against non-interactive traffic, our analysis here applies only to IBW and ICBW, but as we demonstrate experimentally, our attack will work on DSSS watermarks as well.

3.1 Model of Interactive Traffic

We first present a model for interactive traffic, as it is essential to our analysis. Let f_m denote the m-th flow in a pool of interactive traffic flows. Given that the traffic might be encrypted, we do not consider the content of the packets; likewise, the sizes of packets representing keystrokes are likely to be uniform. We thus consider only the arrival time of the packets in the flow, allowing us to model the flow as a point process.

Suppose we observed packet arrivals at times $t_1 < t_2 < \cdots < t_n$ in a fixed interval $(0, \tau]$ such that t_i is the time the i-th packet arrived. The collection of arrival times $\mathbf{t}_m = (t_1, t_2, \ldots, t_n)$ specifies a flow f_m. Furthermore, we model the interactive connection as a Markov-modulated Poisson process (MMPP) [14, 15]. The set of possible states are $\{0, 1\}$, where state 0 corresponds to user typing characters and state 1 corresponds to periods of silence. Figure 8 depicts this two-state MMPP.

Let $X(t)$ denote the state of the process at time t. When the process is in state 0, packet arrivals are modeled as a renewal process; i.e. the interarrival times are independent and identically distributed (i.i.d.). In case of interactive traffic flow, this renewal process is often modeled as Poisson [12, 5]. The Poisson assumption means that the interarrival times of the packets, denoted by θ, are exponentially distributed. Hence their probability density function (PDF) is given by:

$$f_\theta(t) = \lambda e^{-\lambda_0 t}$$

where λ_0 denotes the rate of the Poisson process. When the process is in state 1, the arrivals are again modeled as Poisson but with rate $\lambda_1 < \lambda_0$. Given that state 1 corresponds to a period of silence (no packet arrivals), as soon as a packet arrives, the embedded Markov chain transitions to state 0. Therefore, the transition probabilities $\{P_{ij}, i, j = 0, 1\}$ of the embedded Markov chain $\{X_n, n \geq 0\}$ are as follows:

$$\begin{aligned} P_{00} + P_{01} = 1, \\ P_{01} = 1, P_{11} = 0 \end{aligned} \quad (1)$$

and the embedded Markov chain is defined by the matrix:

$$\begin{bmatrix} P_{00} & 1 \\ 1 - P_{00} & 0 \end{bmatrix}$$

The steady state probabilities π_0, π_1 of the embedded chain X_n are given by:

$$\begin{bmatrix} \pi_0 \\ \pi_1 \end{bmatrix} = \begin{bmatrix} P_{00} & 1 \\ 1 - P_{00} & 0 \end{bmatrix} \begin{bmatrix} \pi_0 \\ \pi_1 \end{bmatrix}$$

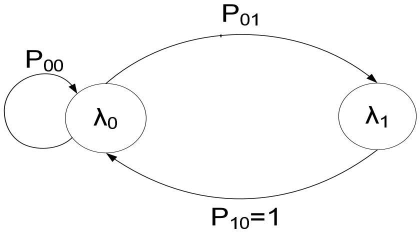


Figure 8: The embedded two-state Markov chain.

or:

$$\pi_0 = \frac{1}{2 - P_{00}}, \qquad \pi_1 = \frac{1 - P_{00}}{2 - P_{00}}$$

The steady state probabilities P_0, P_1 of the Markov process $X(t)$ are given by [15]:

$$P_i = \frac{\frac{\pi_i}{\lambda_i}}{\sum_k \frac{\pi_k}{\lambda_k}}$$

or:

$$P_0 = \frac{\lambda_1}{\lambda_1 + (1 - P_{00})\lambda_0}, P_1 = \frac{(1 - P_{00})\lambda_0}{\lambda_1 + (1 - P_{00})\lambda_0} \quad (2)$$

The significance of the steady state probabilities of (2) is that they capture the probability of each of the states 0 and 1 at any given point in time. Recall that ICBW encodes the watermark bits “1” or “0” by delaying the arrival times of the packets in the set of intervals A_i or B_i respectively and IBW encodes the watermark bits “0” or “1” by transferring the traffic of an interval of length T to some adjacent interval. Therefore, they both create periods of times with no arrivals in the flow. This period for ICBW is of length a and for IBW is of length T. When the embedded Markov chain is in state i, we can compute the probability of zero occurring in a period of length ℓ starting at any given point as:

$$P_{f_m^i}(0; \ell) = e^{-\lambda_i \ell} \quad (3)$$

since the waiting times are exponentially distributed and therefore memoryless.

In general, given a flow f_m generated from an MMPP, from (3), the probability of having a period of length ℓ with no arrivals $P_{f_m}(0; \ell)$ is:

$$\begin{aligned} P_{f_m}(0; \ell) &= P_0 P_{f_m^0}(0; \ell) + P_1 P_{f_m^1}(0; \ell) \\ &= P_0 e^{-\lambda_0 \ell} + P_1 e^{-\lambda_1 \ell} \end{aligned} \quad (4)$$

where the steady state probabilities $\{P_0, P_1\}$ are given by (2).

A good watermarking scheme requires that the watermarked stream should not reveal any clues of the presence of the watermark to unauthorized observer. Therefore, it is desirable to pick ℓ such that $P_{f_m}(0; \ell)$ above should be reasonably large, so that presence of silent periods does not give away the watermark. We next present parameters of our two-state MMPP and show that, for those parameters, the watermark indeed cannot be detected by observing a single stream watermarked with ICBW or IBW. However, we will show that if attackers have access to multiple copies of a marked signal, they can defeat the two watermarking schemes both when multiple flows are watermarked with the same message and when different messages are embedded in different flows.

3.2 Parameter Selection and Goodness of Fit

We estimated the parameters P_{00}, λ_0, and λ_1 of our MMPP model by using network traces of SSH connections taken at a wireless access point in our institution. For a trace, we first estimated the underlying state of the embedded Markov chain by choice of a threshold η. If the interarrival time between two packets exceeded the threshold η, we assumed that the process was in state 1 and if the interarrival time between two packets was less than the threshold η, we assumed that the user was typing and therefore the process was in state 0. Once the states $\{X_n, n \geq 0\}$ of the underlying chain are determined, by concatenation of the parts of the interactive traffic that came from same underlying state, we could extract two Poisson flows with rates λ_0 and λ_1 from the original flow.

Given that the expected number of arrivals of a Poisson process distribution with parameter λ in time interval $(0, t]$ is λt, we estimated the rates λ_0 and λ_1 by calculating the arrival rates of each of the two extracted flows. Parameter P_{00} was estimated as the portion of the time the chain spent at state 0. Our estimated values for the transition probability P_{00} and the rates λ_0 and λ_1 were as follows:

$$P_{00} = .96 \qquad \lambda_0 = 5.6 \qquad \lambda_1 = 0.57 \tag{5}$$

To assess the goodness of fit of our MMPP model with parameters of (5), we used a quantile–quantile (q–q) plot [7]. Using the theoretical CDF of the model, the observations are mapped into values in interval $[0, 1]$. If the underlying statistical model of the data is consistent with the observations, the values obtained from the mapping are uniformly distributed in the interval $[0, 1]$. To assess the uniformity of the mapped values or equivalently assessing the goodness of the theoretical model an empirical CDF of the mapped values is compared against the theoretical CDF of a uniform distribution, which is a 45-degree reference line. The closer the CDF to this reference line, the greater the evidence that the statistical model captures the underlying phenomenon. The q–q plot in Figure 9 shows that the MMPP model for the interactive traffic with parameters (5) provides a good fit for the data and significantly outperforms a simpler Poisson model, or a Pareto distribution that has been previously proposed to fit interactive traffic [19].

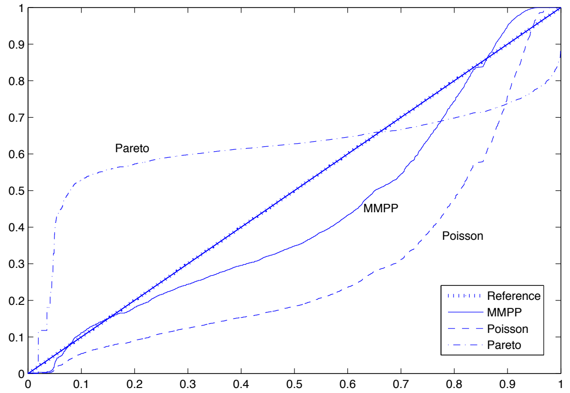


Figure 9: Q–Q plot of Poisson and MMPP models with our sample data.

3.3 Multi-Flow Attack

Regardless of whether the ICBW or IBW watermarking schemes are implemented using the same message across all interactive flows or they use multiple messages for different flows, they are subject to an averaging attack. This is because both schemes embed watermarks by emptying the same parts across various flows. Next, we will explain our attack for both the single-message and multiple-message watermarks.

3.3.1 Single-Message Watermarks

When ICBW or IBW watermarking schemes are implemented using the same message across all interactive flows, an attacker who has access to k watermarked flows can form an aggregate of all the flows, taking the sorted union of all the arrival times of packets in all flows. We denote this aggregated stream by $\overline{f_k}$, where the subscript k denotes the number of streams involved in forming the aggregate flow.

Given that each interactive stream is independent of all the other streams, the probability of having a period of length T with no arrivals in the flow $\overline{f_k}$ is given by:

$$\begin{aligned} P\{N_{\overline{f_k}}(t_a + \ell) - N_{\overline{f_k}}(t_a) = 0\} &= \prod_{i=1}^{k} P_{f_i}(0; \ell) \\ &= P_{f_m}(0; \ell)^k \end{aligned} \tag{6}$$

Equation (6) shows that probability of having period of length ℓ with no arrivals decreases exponentially in k, the number of copies used to form the aggregate flow $\overline{f_k}$. Therefore, if the streams are not watermarked there is a very small probability that the aggregate stream has periods of no arrivals. However, if both ICBW and IBW use the same key and message across all interactive flows, the aggregated copy of the watermarked flows always exhibits patterns of no arrivals of length ℓ that give away the location of the watermark as well as the maximum delay parameter a of ICBW and the period T of IBW.

Substituting the parameters of (5) into (4), assuming $\ell = 350ms$, as suggested by Wang et al. [25], we have $P_{f_m}(0; 0.35) = 0.33$. Therefore, in an aggregate of as few as 10 flows probability of a period of 350ms without any arrivals is as low as $P_{f_m}(0; 0.35)^{10} = 1.6 \times 10^{-5}$. Similarly, for $\ell = 900ms$, as used by Pyun et al. [21], we have $P_{f_m}(0; 0.9) = 0.17$ and $P_{f_m}(0; 0.9)^{10} = 2.4 \times 10^{-8}$.

3.3.2 Multi-Message Watermarks

If different flows are used to encode different messages, simple aggregation will no longer work, since by switching between 1 and 0 bits, both ICBW and IBW apply different transforms to different intervals. For example, with ICBW, a given interval may be squeezed when a certain bit is 0, and not squeezed when that bit is 1. By aggregating flows where that bit changes, no empty periods will be detected.

However, by observing a few more flows, we can still detect the presence of a watermark. Given a bit b and a set of $2k-1$ flows, by the pigeon hole principle, there exists a subset of k flows where the bit has the same value. If we aggregate all the flows in that subset, we will find clear intervals of length a or T, depending on the scheme that is used.

To detect the watermark, then, we examine all $\binom{2k-1}{k}$ subsets of k flows out of a collection of $2k-1$. For each bit position, we will be able to find at least one subset where that bit value is all the same, and we can thus detect it with the same ease as when a single-valued watermark is used. The number of subsets is, of course, superexponential in k, but our attack works with values of k around 10, making such a search feasible, as $\binom{19}{10} = 92378$.

Examining all these subsets increases the possibility of a false positive—a naturally occurring cleared interval in the aggregate flow. However, such false positives will be relatively rare, so the attacker can estimate the value of ℓ and then discard intervals that do not match.

3.4 Impact of Timing Perturbations

Our analysis so far has assumed that the attacker sees the timings of the watermarked stream directly. In reality, these timings will be perturbed by network delays. As a result, the intervals cleared by the watermark may have some packets from previous intervals shifted into them and no longer appear completely empty. Note that what is relevant here is not the magnitude of the network delay but its variance, or *jitter*, since delaying all packets by an equal amount does not affect our attack. And if the jitter is much less than ℓ, our attack will work equally well: if jitter is $< \epsilon$ with high probability, then we will find clear intervals of length at least $\ell - \epsilon$ in the k aggregated watermarked streams, whereas the probability of seeing such an interval in unwatermarked streams is $P_{f_m}(0; \ell-\epsilon)^k \approx P_{f_m}(0; \ell)^k$, which is vanishingly small. We observe that the studied parameters of the ICBW and IBW schemes have ℓ = 350ms or 900ms, in order to resist traffic perturbations, repacketization, etc. The network jitter, on the other hand, is two orders of magnitude smaller. Our experiments on PlanetLab [3] show it to be on the order of several milliseconds for geographically distributed hosts, and this matches the results of previous studies [18]. Therefore, it is indeed the case that the jitter is $< \epsilon \ll \ell$, so it will not significantly affect our attack.

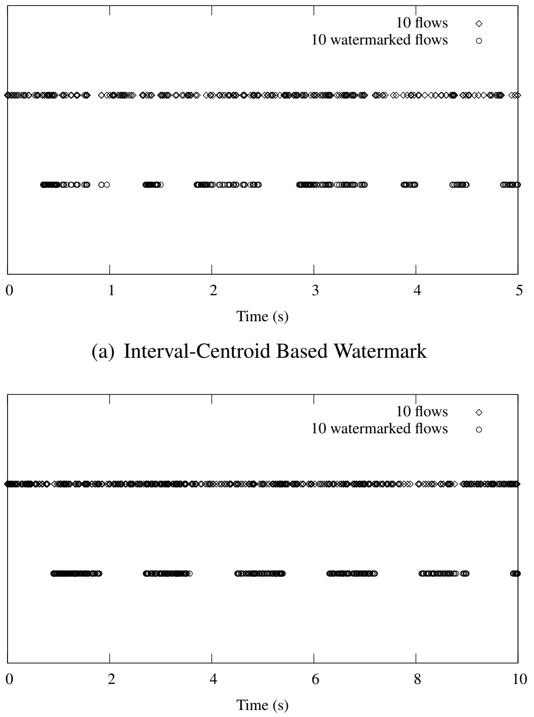
4 Implementation

Having shown the theoretical background behind our attack, we now show the result of implementing it in practice. We developed algorithms to detect the presence of a watermark, recover the secret parameters, and to remove the watermark from new streams. We evaluated the algorithms using both real flows gathered from traces and synthetic flows generated using our MMPP model, presented in Section 3.1. We first present our attacks for single-message watermarks, and then extend it to watermarks that use multiple messages.

4.1 Watermark Detection

As above, our attack relies on collecting a series of flows that are watermarked with the same message. These flows are combined into a single flow and examined for large gaps between packets. Figure 10(a) shows the packet arrivals for 10 combined flows before and after an ICBW watermark has been applied. The watermark pattern is clearly visible in the combined flows, revealing the presence of a watermark. Figure 10(b) shows the same process working with the IBW watermark scheme.

We also performed the same analysis for non-interactive, bulk transfer traffic by applying the watermark to packet traces we collected from web downloads



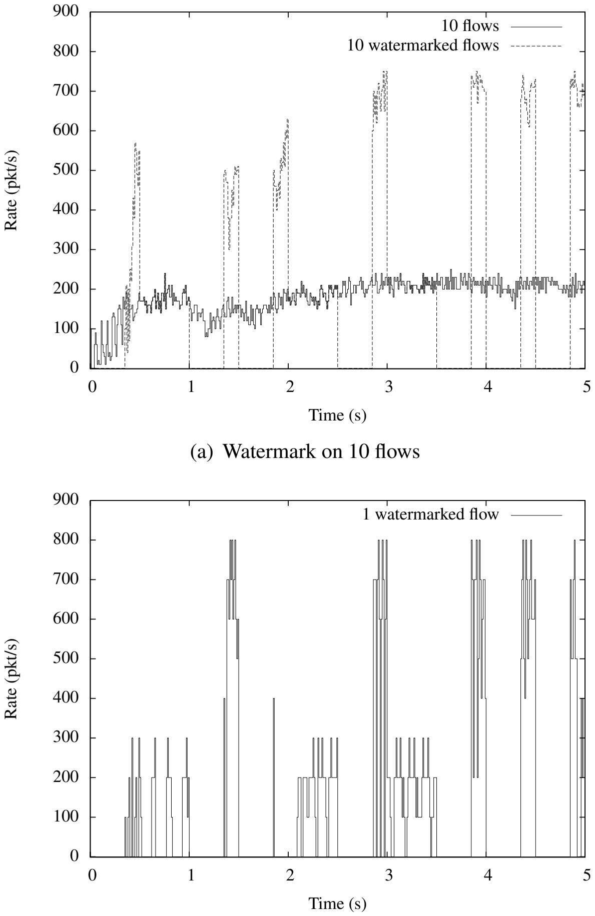
(a) Interval-Centroid Based Watermark

(b) Interval-Based Watermark

Figure 10: 10 flows before and after watermarking.

across a DSL connection. Figure 11(a) shows the packet timings for 10 combined flows before and after a watermark. Bulk transfers have a somewhat more regular behavior, since they are controlled by the TCP algorithms, rather than by individual users. This can be seen at the beginning of the 10 combined flows before watermark: the TCP slow start period results in a much lower rate for the first few seconds of the connection. However, this regularity quickly gets out of sync due to irregular network delay and response times. In the graph of 10 watermarked flows, the intervals squeezed by the watermark are readily visible. In fact, because data transfer flows are much more dense than interactive flows, the watermark is visible even on a single flow (Figure 11(b)).

The DSSS watermark is intended to be applied to bulk transfer traffic such as FTP, since it interferes with traffic rate, rather than changing packet timings. A similar multi-flow attack works against DSSS as well, as shown in Figure 12. (We used the parameters of chip length 0.4s, chip sequence length of 7, and code length of 7.) In this case, periods of high interference are clearly seen as low-rate periods in the flows, allowing one to recover the chip sequence and then decode the watermark.



(a) Watermark on 10 flows

(b) Watermark on 1 flow

Figure 11: Watermark detection on bulk traffic.

4.2 Watermark Removal

Based on the combined graphs, it is easy to recover the watermark parameters as well. We can build a template of clear intervals by selecting all intervals larger than a threshold; for example, Figure 13(a) shows the template derived from 10 flows watermarked by ICBW. The estimated template is somewhat imprecise, due to network jitter, as well as the fact that small (10–20ms) gaps may precede or follow the clear intervals even when 10 flows are combined. However, this imprecision is not a problem since the watermark can still be effectively removed. The template also lets us estimate the values of T and a. We can average the lengths of clear intervals and the distance between two consecutive clear intervals to obtain a relatively precise estimate. Armed with this information, we can then modify a new flow to remove the watermark.

For ICBW, we have two choices: we can either shift traffic into the clear intervals in the template, thereby negating the squeezing action of the watermark, or find

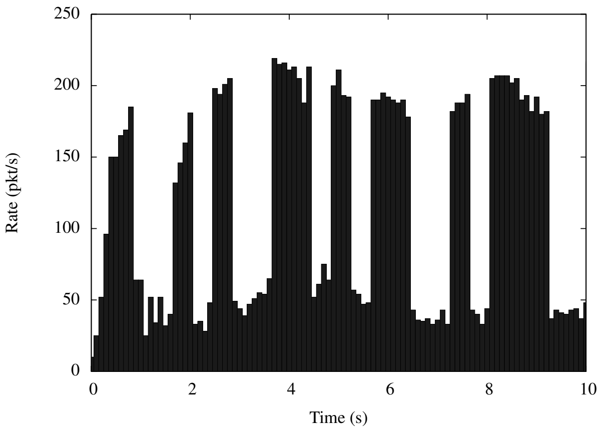
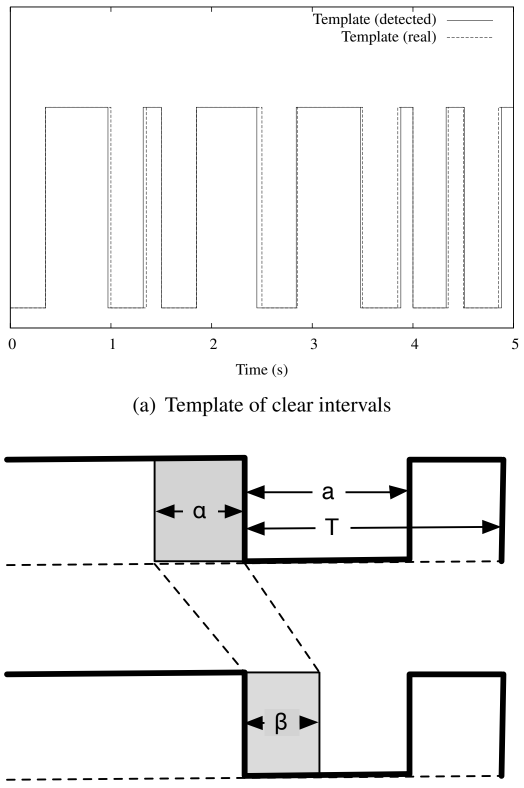


Figure 12: Average rate of 10 flows after DSSS watermark.

intervals that have not been squeezed and squeeze them. We decided to implement the former approach since it does not require as precise an estimate of T. Also, it leaves the flow looking more natural. Our shift is implemented as shown in Figure 13(b), by shifting all packets in a period α before the clear interval into an interval of length β inside the clear interval. Larger values of α and smaller values of β will more significantly shift the interval centroid back in a different direction; however, very small values of β may not have the desired effect, since the template is imprecise and too many packets may get shifted without arriving into the correct interval. Experimentally, we found that $\alpha = 0.9(\hat{T} - \hat{a})$ and $\beta = 0.8(\hat{T} - \hat{a})$ provide best results, where $\hat{T}$ and $\hat{a}$ are estimated values of T and a.

Table 1 shows the results of watermark removal. We reimplemented the ICBW detection mechanism and computed the Hamming distance of the encoded watermark to the detected one, collected over 100 flows. (We show the average distance, with range shown in parentheses). With as few as 10 flows, we are able to get a reasonably good estimate of T and a and remove the watermark in most cases—the ICBW detection scheme uses a Hamming distance threshold of 5–8 to decide when a watermark has been detected. With 15 flows, we get a more accurate template and estimate, and all 100 flows will clear the template.

A similar approach can be used to attack the IBW watermark; by delaying packets so that they fall into the clear intervals, the clear intervals become indistinguishable from loaded ones. Table 2 shows the effect of applying our attack on the IBW watermark, where 24 bits are encoded at different levels of redundancy. Even with a redundancy of 80, most bits are not recovered correctly. These results were obtained by using the code provided by the authors of [21].



(a) Template of clear intervals

(b) Shift to remove watermark

Figure 13: Watermark Removal

We expect a similar technique should work against DSSS watermarks; a template of low rates can be inferred from several flows. An attacker can then decrease rates in the non-interference section of the template by dropping packets, or increase the rate in the high-interference section by delaying packets into the template. We do not have experimental results for DSSS since the detection algorithm is fairly complex and we did not have access to an implementation of it.

4.3 Multiple Messages

So far we have assumed that the watermarks on all of the aggregated flows are the same. Here, we consider the case where each watermark uses different messages. As described in Section 3.3.2, we can still execute our attack by relying on the fact that within a collection of $2k - 1$ flows, for any given bit b, we can find k flows where this bit has the same value.

Figure 14(a) plots the result of such a subset search. By inspection, we can see that in the first subset of flows, the interval (4.5,4.85) has been cleared. In the second subset, this interval remains cleared and the interval (0,0.35) becomes clear as well. The third subset

Table 1: Results for removing ICBW watermarks

Num flows	$\hat{a}$	$\hat{T}$	Hamming not watermarked	Hamming watermarked	Hamming attacked	Ave. delay	Max delay
10	365 ($\sigma = 10.7$)	492 ($\sigma = 15.2$)	17.9 (13–24)	2.67 (1–7)	13.9 (2–20)	33.6	164
15	353 ($\sigma = 0.60$)	504 ($\sigma = 1.62$)	17.6 (13–25)	2.74 (0–6)	16.1 (12–21)	42.6	188.2
20	346 ($\sigma = 0.30$)	504 ($\sigma = 0.50$)	17.2 (12–21)	2.68 (0–5)	16.4 (11–20)	45.4	194.3

Table 2: Watermark bits detected before and after applying the attack (watermark length is 24).

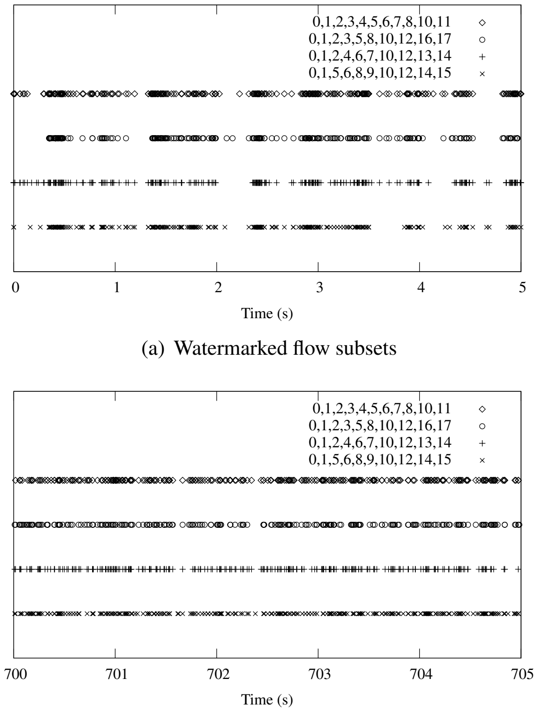
Rep.	Bits detected		Marked packets
	Before attack	After attack	
1	7	3	53
5	14	5	156
10	24	4	505
15	24	2	754
20	24	2	967
24	24	2	1209
30	24	2	1440
35	24	2	1724
41	24	2	2008
45	24	2	2307
50	24	2	2697
55	24	2	3083
60	24	2	3296
65	24	2	3623
70	24	2	3876
75	24	2	4090
80	24	2	4343

has no packets in (2.0,2.35) and the fourth in (3.5,3.85). Note that this pattern immediately lets us detect the presence of a watermark; Figure 14(b) shows the same flow subsets on an unwatermarked section.

Recovery of the secret parameters can proceed largely as in the single-message case. One difficulty is that with the flow subsets, we may encounter large intervals that are not precisely aligned with the interval positions. For example, Table 3(a) lists the blank intervals longer than 0.2s in the last subset. There are a lot of wrong-size intervals that result from the case when 8 or 9 of the flows in the subset have had an interval squeezed, but the last one or two add a few packets to the mix. To address this concern, we can select the largest empty intervals in any subset, as shown in Table 3(b). These will correspond to intervals that have been squeezed on every flow. This can be used to recover the watermark parameters of T and a.

Once these are obtained, the next step is to scan through all subsets and determine which intervals are always squeezed at the same time and call such lists S_i; these will correspond to either A_b or B_b for some bit b. Then, for each S_i, we find S_j such that S_i and S_j are never squeezed at the same time. This will tell us that S_i and S_j correspond to the same bit. Armed with this knowledge, we can remove the watermark by observing the watermarked stream for a short while, and when we see intervals from S_i that are being squeezed, we proceed to artificially squeeze intervals in S_j (or unsqueeze further intervals in S_i, or both).

Note that the subset technique can also be applied when not all the flows are watermarked. For example, a website may watermark only some connections that are



(a) Watermarked flow subsets

(b) Un-watermarked flow subsets

Figure 14: Subset approach to multiple message watermarks

Table 3: Blank intervals from subset of flows

(a) All blank intervals

Start	End
2.08	2.32
3.50	3.85
4.03	4.25
5.13	5.33
11.59	11.85
18.14	18.37
19.56	19.79
25.58	25.82
30.06	30.34
34.08	34.35
...	...

(b) Largest blank intervals

Start	End
130.98	131.35
140.49	140.86
151.99	152.36
161.99	162.35
235.99	236.37
306.49	306.86
334.49	334.86
368.49	368.86
43.99	44.36
51.98	52.35
...	...

of particular interest; by finding subsets that are all watermarked, the mark can still be recovered. A scheme that probabilistically marked some flows and used different messages at the same time would present a challenge to our attack; however, we suggest that a different countermeasure be used, since it allows all flows to be marked, which is desirable for most applications.

5 Countermeasures

We next consider several countermeasures to our attack.

5.1 Multiple Offsets

The watermarking schemes we analyze have the ability to self-synchronize by trying different values for the offset o and using the best match. Thus, o can be changed for different streams. The synchronization mechanism can introduce more errors into the detection, but the use of increased encoding redundancy can make up for it.

The use of different offsets makes our attack more difficult, since simply aggregating k flows will result in misalignment, destroying the clear intervals. It is, of course, possible to test different positions for o for each stream, but to test n positions in k flows requires n^{k-1} trials (we can hold the first flow fixed).

On the other hand, some alignments of two or three flows can be discarded immediately, if such an alignment results in few intervals that are clear of packets. Furthermore, the search for o can be imprecise at first: even if each flow is aligned to within 0.1s of the correct position, intervals of 150ms or 700ms will be seen in the average. Thus, changing offsets makes our attack more difficult, but not impossible to perform.

5.2 Multiple Positions

Another alternative is to choose different positions, in the case of ICBW and IBW, and different PN codes in the case of DSSS. Let us consider the case of ICBW. A watermarker and detector must use the same assignment of intervals to the sets A_i and B_i, as determined by the random seed s, in order for the watermark to be successfully recovered. However, a watermarker may decide to use multiple seed values, $s_1, \ldots, s_n$, and pick one of them at random for each flow.

To deal with this, the detector would need to try to recover the watermark with each possible s_i and pick the best match. Once again, the probability of error grows with n, but increased redundancy can again be used to make up for it. Note that the probability of error falls exponentially with increased redundancy, but grows only roughly linearly with n.

We can once again use the subset attack to try to find k flows that use the same seed value s_i; however, the complexity grows quickly out of control. The probability of a given set of k flows using the same seed is $\left(\frac{1}{n}\right)^{k-1}$, which falls quite quickly even when $k = 10$. By the pigeon hole principle, within $n(k-1)+1$ flows we can always find a subset of k flows with the same seed, but the search space of all $\binom{n(k-1)+1}{k}$ subsets grows super-exponentially in n. For example, with $n = 6$ and $k = 10$, $\binom{51}{10} > 10^{10}$, resulting in an infeasible number of subsets to enumerate.

The same principle can apply to IBW, by picking multiple sets of positions $\{s_i\}$, and to DSSS by using multiple PN codes.

6 Conclusion

We have demonstrated an attack on the interval centroid-based watermarking scheme and interval based watermarking scheme that is highly successful, while requiring a low amount of resources. Our attack is based on a solid theoretical grounding, and has been validated with a prototype implementation tested against the original ICBW and IBW prototypes. We can remove the watermark from an existing flow for both schemes. Additionally, in case of IBW we can also recover the watermark parameters and values, allowing us to modify the watermark or insert it into other streams, thereby confusing the detector. We have also suggested a countermeasure to our attack—switching bit positions. This countermeasure can impose a very high computation cost and therefore disable the attack.

While the use of network flow watermarking techniques for various security applications is quite new [27, 21, 25, 28], digital watermarking. and specifically multimedia watermarking is a nearly mature field. Indeed, most of network flow watermarking schemes are inspired by multimedia watermarks. To name a few,, Wang and Reeves's [27] scheme is a special instance of QIM watermarking, a well-understood multimedia watermarking technique [16]. The IBW scheme of Pyun et al. [21] is based on the patchwork watermark of Bender et al. [4] and the scheme of Yu et al. [28] is based on spread spectrum watermarking [9].

The current approach for designing network flow watermarks suffers from the fact that. while watermarking schemes are inspired by the digital watermarking schemes, little attention is given to the entirety of the watermarking design problem. For example, statistical characteristics of the underlying media are always an important consideration in digital watermarks, but network watermark research does not adequately model the effect that network traffic characteristics have on watermarks; as we showed, the density of bulk traffic makes it very difficult to insert a transparent watermark. Likewise, digital watermarks have long considered the possibility that multiple watermarked documents can be used to attack watermarks [10, 17], but we are unaware of previous work looking at the multi-flow threat model for watermarking. We thus hope that future work on watermarks will be informed by our work and perform a broader analysis.

Acknowledgments

We would like to thank Peng Ning and Douglas Reeves for providing us with the IBW implementation, and Xinyuan Wang and the anonymous referees for providing feedback on earlier versions of this paper. This research was supported in part by NSF grants CNS–0627671 and CCF–0729061.

References

[1] BACK, A., GOLDBERG, I., AND SHOSTACK, A. Freedom systems 2.1 security issues and analysis. White paper, Zero Knowledge Systems, Inc., May 2001.

[2] BACK, A., MÖLLER, U., AND STIGLIC, A. Traffic analysis attacks and trade-offs in anonymity providing systems. In *Information Hiding Workshop* (Apr. 2001), I. S. Moskowitz, Ed., vol. 2137 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 245–247.

[3] BAVIER, A., BOWMAN, M., CHUN, B., CULLER, D., KARLIN, S., MUIR, S., PETERSON, L., ROSCOE, T., SPALINK, T., AND WAWRZONIAK, M. Operating systems support for planetary-scale network services. In *Symposium on Networked Systems Design and Implementation* (Mar. 2004), R. Morris and S. Savage, Eds., USENIX, pp. 253–266.

[4] BENDER, W., GRUHL, D., MORIMOTO, N., AND A.LU. Techniques for data hiding. *IBM Systems Journal 35*, 3/4 (1996), 313–336.

[5] BLUM, A., SONG, D. X., AND VENKATARAMAN, S. Detection of interactive stepping stones: Algorithms and confidence bounds. In *International Symposium on Recent Advances in Intrusion Detection* (Sept. 2004), E. Jonsson, A. Valdes, and M. Almgren, Eds., vol. 3224 of *Lecture Notes in Computer Science*, Springer, pp. 258–277.

[6] BOYAN, J. The anonymizer: Protecting user privacy on the web. *Computer-Mediated Communication Magazine 4*, 9 (September 1997).

[7] BROWN, E. N., BARBIERI, R., VENTURA, V., KASS, R. E., AND FRANK, L. M. The time-rescaling theorem and its application to neural spike train data analysis. *Neural Computation 14*, 2 (2002), 325–346.

[8] CHAUM, D. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM 24*, 2 (February 1981), 84–90.

[9] COX, I., KILIAN, J., LEIGHTON, T., AND SHAMOON, T. Secure spread spectrum watermarking for multimedia. *IEEE Transactions on Image Processing 6*, 12 (1997), 1673–1687.

[10] COX, I. J., KILLIAN, J., LEIGHTON, T., AND SHAMOON, T. Secure spread spectrum watermarking for images, audio, and video. In *IEEE International Conference on Image Processing* (1996), vol. III, pp. 243–246.

[11] DINGLEDINE, R., MATHEWSON, N., AND SYVERSON, P. Tor: The second-generation onion router. In *USENIX Security Symposium* (Aug. 2004), M. Blaze, Ed., USENIX Association, pp. 303–320.

[12] DONOHO, D., FLESIA, A., SHANKAR, U., PAXSON, V., COIT, J., AND STANIFORD, S. Multiscale stepping-stone detection: detecting pairs of jittered interactive streams by exploiting maximum tolerable delay. In *International Symposium on Recent Advances in Intrusion Detection* (Oct. 2002), A. Wespi, G. Vigna, and L. Deri, Eds., vol. 2516 of *Lecture Notes in Computer Science*, Springer, pp. 16–18.

[13] FEDERRATH, H., Ed. *International Workshop on Design Issues in Anonymity and Unobservability* (July 2000), vol. 2009 of *Lecture Notes in Computer Science*, Springer.

[14] FISCHER, W., AND MEIER-HELLSTERN, K. The Markov-modulated Poisson process (MMPP) cookbook. *Performance Evaluation 18*, 2 (1993), 149–171.

[15] GALLAGER, R. G. *Discrete Stochastic Processes*. Kluwer Academic Publishers, 1996.

[16] GOTETI, A. K., AND MOULIN, P. QIM watermarking games. In *IEEE International Conference on Image Processing* (2004), pp. 717–720.

[17] KILIAN, J., LEIGHTON, F., MATHESON, L., SHAMOON, T., TARJAN, R., AND ZANE, F. Resistance of digital watermarks to collusive attacks. In *IEEE International Symposium on Information Theory* (1998), p. 271.

[18] MARSH, I., AND LI, F. Wide area measurements of VoIP quality. In *Workshop on Quality of Future Internet Services* (2003).

[19] PAXSON, V., AND FLOYD, S. Wide-area traffic: The failure of Poisson modeling. *IEEE/ACM Transactions on Networking 3*, 3 (June 1995), 226–244.

[20] PFITZMANN, B., AND MCDANIEL, P., Eds. *IEEE Symposium on Security and Privacy* (May 2007).

[21] PYUN, Y., PARK, Y., WANG, X., REEVES, D. S., AND NING, P. Tracing traffic through intermediate hosts that repacketize flows. In *IEEE Conference on Computer Communications (INFOCOM)* (May 2007), G. Kesidis, E. Modiano, and R. Srikant, Eds., pp. 634–642.

[22] RAYMOND, J.-F. Traffic analysis: Protocols, attacks, design issues, and open problems. In Federrath [13], pp. 10–29.

[23] SYVERSON, P., TSUDIK, G., REED, M., AND LANDWEHR, C. Towards an analysis of onion routing security. In Federrath [13], pp. 96–114.

[24] WANG, X., CHEN, S., AND JAJODIA, S. Tracking anonymous peer-to-peer VoIP calls on the Internet. In *ACM Conference on Computer and Communications Security* (Nov. 2005), C. Meadows, Ed., ACM, pp. 81–91.

[25] WANG, X., CHEN, S., AND JAJODIA, S. Network flow watermarking attack on low-latency anonymous communication systems. In Pfitzmann and McDaniel [20], pp. 116–130.

[26] WANG, X., REEVES, D., AND WU, S. F. Inter-packet delay based correlation for tracing encrypted connections through stepping stones. In *European Symposium on Research in Computer Security* (Oct. 2002), D. Gollmann, G. Karjoth, and M. Waidner, Eds., vol. 2502 of *Lecture Notes in Computer Science*, Springer, pp. 244–263.

[27] WANG, X., AND REEVES, D. S. Robust correlation of encrypted attack traffic through stepping stones by manipulation of interpacket delays. In *ACM Conference on Computer and Communications Security* (2003), V. Atluri, Ed., ACM, pp. 20–29.

[28] YU, W., FU, X., GRAHAM, S., D.XUAN, AND ZHAO, W. DSSS-based flow marking technique for invisible traceback. In Pfitzmann and McDaniel [20], pp. 18–32.

[29] ZHANG, Y., AND PAXSON, V. Detecting stepping stones. In *USENIX Security Symposium* (Aug. 2000), S. Bellovin and G. Rose, Eds., USENIX Association, pp. 171–184.

Notes

[1] We are unaware of a quantitative comparison of the accuracy of watermarking techniques with passive traffic analysis, but reported false-positive rates for most watermarking techniques are quite low. In any case, the two techniques can be combined to improve accuracy.

[2] Yu et al. suggest that this can be done by sending an interfering flow across a bottleneck link; their scheme is thus unique in not requiring full control of packet forwarding for the flow.