DMark: A Data-dependent Network Flow Watermark

Amir Houmansadr* Nikita Borisov*
*Dept. of Electrical and Computer Engineering
University of Illinois at Urbana–Champaign
{ahouman2,nikita}@illinois.edu

Abstract

Flow watermarks are active traffic analysis techniques that help to establish a causal connection between two network flows. As compared with passive traffic analysis, they can provide much more efficient detection and lower error rates, and also lower oder of computation in the case blind watermarking. To be useful, a watermark must be made to be robust to lost and reordered packets; however, the blind interval-based techniques to achieve this have been shown to be susceptible to a multi-flow attack that can be used by an adversary to find watermarks in flows and remove them.

We develop a new blind network flow watermarking scheme, DMark, that is robust to network perturbations, e.g., packet losses and delays , but that is invisible and immune to multi-flow attacks. Our approach is to use a data-dependent marking strategy, such that each flow is marked with a different pattern. DMark introduces very small perturbations to network traffic; this makes it difficult to detect by attackers and also makes it transparent in enterprise installations. DMark can be detected very efficiently; therefore, it can also be used to perform large-scale traffic analysis of anonymous communication systems. We perform both a mathematical analysis of DMark as well as an evaluation using the PlanetLab testbed. Our findings show that DMark can achieve very low false error rates, even in the presence of significant network perturbations, while providing watermark invisibility and robustness to multi-flow attack.

1 Introduction

Network intruders usually try to hide their real location by relaying their traffic through a number of intermediate hosts, called *stepping stones* [20]. The traffic is encrypted, preventing simple identification of stepping stones; instead, traffic analysis techniques are used to find flows that have similar characteristics, based on features such as packet timings, counts, and sizes [14, 20, 18, 16, 6, 4]. Traffic analysis can also be used to attack anonymous communication systems by finding relationships between two flows that would otherwise be unlinkable [1, 5, 12].

An alternative traffic analysis technique is to actively perturb traffic features, rather than passively observing them, by inserting a network flow *watermark* [17, 19, 15, 11, 7]. If a watermark inserted into one flow is later detected in another, this indicates a causal relationship between the two flows. Compared to passive schemes, watermarks can achieve higher detection accuracies on shorter flows. Most watermark schemes also perform *blind* detection, where the watermarker and detector share only a secret key but do not exchange information about flows; this reduces the communication overhead to $O(1)$ and the computation overhead to $O(n)$ in the number of flows, as compared to $O(n)$ and $O(n^2)$ for passive and non-blind schemes. This allows efficient detection of stepping stones in very large organizations and ISPs; likewise, watermarks can be used for large scale traffic analysis of anonymous systems, whose populations can reach hundreds of thousands of users [10].

To be effective, watermarks must be robust to perturbations in network flows, such as lost or retransmitted packets. Recent watermarks have used an interval-based approach to achieve this: rather than affecting individual packets, they split the flow into time intervals and apply transformations to an entire interval to insert a mark [11, 15, 19]. This makes the mark less sensitive to changes in individual packets. However, Kiyavash et al. showed that interval-based techniques are subject to a multi-flow attack (MFA): multiple

flows that are watermarked can be lined up to find transformed intervals and recover the secret watermark parameters. This could be used by stepping stones or anonymous routers to detect watermark presence and effect countermeasures. Houmansadr et al. proposed an alternate approach to achieving robustness by using sliding-window matching [7]; however, their approach is non-blind and thus has the same scaling limitations as passive traffic analysis techniques.

We propose a new watermarking scheme, DMark, that is both robust to packet losses and reordering and at the same time immune to multi-flow attacks. DMark is based on creating a data-dependent watermark: we change the process of how the watermark is applied based on the features of the network flow, in a sense using flow features to augment the secret watermarking key. As a result, the watermark will appear differently on different flows, rendering the multi-flow attack ineffective. We also make sure that the amount of extra delay introduced by our watermark is minimal; this ensures that a watermark does not introduce undue overhead for network flows, which is an important feature since in enterprise settings, many benign flows would be marked. Small delays also reduce the chance that the presence of a watermark would be detected by an attacker. We develop a mathematical model to analyze false error rates of DMark. In addition, we evaluate DMark using simulation and a testbed evaluation on PlanetLab [2].

The rest of this paper is organized as follows: in Section 2 we describe the design of the DMark watermarking scheme, along with the dependency algorithm in Section 2.1. By modeling the network flow behaviour, DMark is analysed in Section 3 to provide false errors analysis of the scheme, while the choice of system parameters is discussed in Section 4. The results of the DMark simulations are presented in Section 5 and the implementation results over the Internet are discussed in Section 6. Watermark invisibility and robustness to multi-flow attacks are discussed in Section 7, and some suggestions for system extension are proposed in Section 8. Finally, the paper is concluded in Section 9.

2 Data-dependent flow watermarking scheme

In this section we describe the design of the data-dependent network flow watermarking scheme which is based on using a dependency algorithm discussed in the following section.

2.1 Dependency algorithm

A dependency algorithm is used in the design of DMark watermarking scheme in order to provide robustness to multi-flow attacks. The structure of the dependency algorithm is motivated by Shamir's secret sharing scheme [13].

Secret sharing is used to divide a secret S into a number if shares, say n, such that for some $k \leq n$ cooperation of any k or more such shares gives away the secret value S while no group of shares less than k is able to reveal the secret. Shamir's (k, n)-threshold scheme is one of the first secret sharing schemes and is based on the characteristics of polynomials. More specifically, for any secret S to be shared $k-1$ coefficients $a_1, \ldots, a_{k-1}$ are selected randomly to form the following $k-1$-degree polynomial:

$$f(x) = S + a_1 x + \ldots + a_{k-1} x^{k-1} \tag{1}$$

The $n \geq k$ secret shares are then generated by evaluating the polynomial at n different points of the horizontal axis and distributing the pairs $(x_i, f(x_i))$. It can be shown that having any k or more such a shares reveals the secret S while no smaller group of shares can do so. All the calculations of the mentioned Shamir's scheme are done modulo k to provide some desirable cryptographical properties.

We devise a dependency algorithm, $D_{1,1,1}$, which is motivated by Shamir's $(2, 2)$-threshold secret sharing scheme. We represent the dependency algorithm as:

$$S = D_{1,1,1}[(X_1, Y_1), (X_2, Y_2)] \tag{2}$$

where (X_1, Y_1) and (X_2, Y_2) are two distinct points in the Cartesian plane and S is the y-intercept of the straight line containing (X_1, Y_1) and (X_2, Y_2). This results in:

$$S = \frac{X_2 Y_1 - X_1 Y_2}{X_2 - X_1} \tag{3}$$

We call (X_1, Y_1) and (X_2, Y_2) *dependant*s of the *secret* S and we have that $-A \leq Y_1, Y_2, S \leq A$ for some A. In Section 8 we propose to use other notions of the dependency algorithm, $D_{B,M,G}$, where B, M, and G represent number of base intervals, number of mark intervals, and the degree of the polynomial used, respectively.

2.2 Watermark insertion scheme

To watermark a network flow, the watermarker divides it into non-empty non-overlapping intervals of length T of time. To deal with the possible non-empty intervals the watermarker ignores all the empty intervals of length more than $0.75T$, as will be discussed later. Also, an *initial offset* o ranging within $[0, T]$ is randomly selected for each flow to be watermarked in order to determine the start point of the very first interval of that flow. We insert a b-bit watermark, for which watermarker needs to use $2b$ non-empty intervals of the flow. The watermarking scheme relies on relating different intervals using the $D_{1,1,1}$ dependency algorithm.

Before the watermarking process starts, a watermark key is generated and shared between watermarker(s) and watermark detector(s). For any watermark bit, say i^{th} watermark bit, we assign a *secret value* $S(i)$, which is randomly chosen such that:

$$S(i) \in \left[-\frac{T}{2}, \frac{T}{2}\right] \qquad \forall i = 1, \ldots, b \tag{4}$$

In addition, we assign two intervals to any watermark bits, calling the first interval the *base interval*, $base(i)$, and the other interval the *mark interval*, $mark(i)$, of that watermark bit. The main idea of the watermarking is to share the secret value of any watermark bit i, $S(i)$, between the corresponding base and mark intervals of that bit using the mentioned dependency algorithm. Different features of flow intervals are used to represent the dependants of the secret values, as would be described later. This completes generation of the watermark key, K, which is only shared between watermarker(s) and detector(s) of the watermarking scheme:

$$K = \{S(i), base(i), mark(i); \text{ for } i = 1, \ldots, b\} \tag{5}$$

As mentioned above, watermarking is based on having base and mark intervals of each watermark bit represent corresponding dependants of the secret value of that watermarking bit. Different interval features are used for base and mark intervals to represent the secret value shares; for the base interval, the *interval centroid* is used to represent the corresponding dependant. The centroid of an interval is defined as the average of arriving times of all the packets in that interval, normalized to the center of the interval. In other words, the centroid of the i^{th} interval is defined as:

$$C(i) = \frac{1}{pckt(i)} \sum_{p=1}^{pckt(i)} \left(t_p - (i - \frac{1}{2})T\right) \tag{6}$$

where $pckt(i)$ is the number of packets arriving within interval i, and t_p is the arriving time of p^{th} packet of that interval. Clearly, the defined centroid varies within the range $[-\frac{T}{2}, \frac{T}{2}]$. If no packet arrives in some time interval i we do not use that interval for watermarking, as will be discussed more later.

To insert the i^{th} watermark bit, we use the corresponding secret value $S(i)$ and the centroid of its base interval $C(base(i))$ to derive $Y(i)$ so that $(X_2, C(base(i)))$ and $(1, Y(i))$ represent two corresponding dependants of the secret value $S(i)$ using the $D_{1,1,1}$ dependency algorithm with parameter $A = T/2$. In other words:

$$S(i) = D_{1,1,1}[(1, Y(i)), (X_2, C(base(i)))] \tag{7}$$

where $D_{1,1,1}$ is defined in (2). X_2 is a fixed parameter of the watermarking scheme and its value is discussed later in Section 4.1. Figure 1 illustrates how the dependency algorithm is used in the watermark insertion scheme. The $Y(i)$ parameter derived as mentioned above is then used to make certain modifications to the corresponding mark interval, $mark(i)$, which is described in the following, completing insertion of the i^{th} watermark bit.

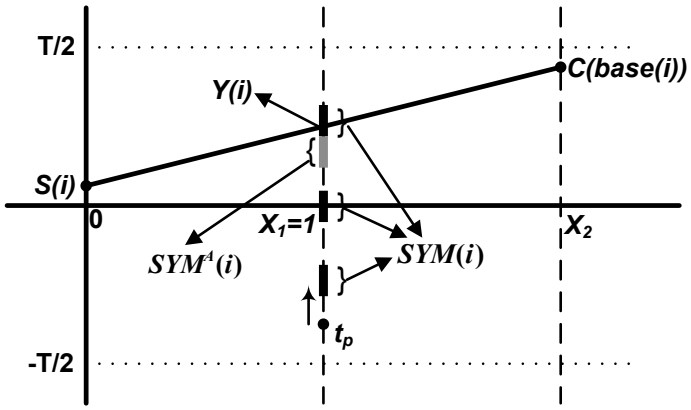


Figure 1: Use of $D_{1,1,1}$ dependency algorithm in the watermark insertion scheme. $SYM(i)$ is the assigned symbol, and $SYM^A(i)$ is the nearest adjacent symbol.

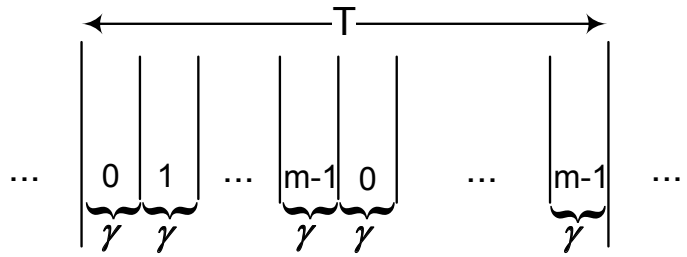


Figure 2: Assignment of symbols to subinterval in a mark interval.

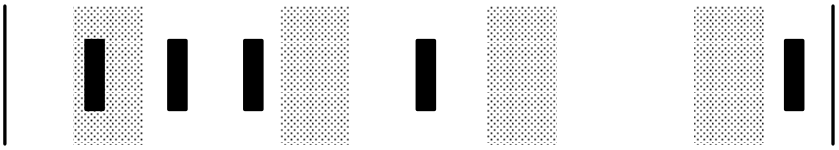
Watermark is inserted by making changes to the mark intervals. Each mark interval is divided into $m \times r$ non-overlapping subintervals of length $\gamma = \frac{T}{m \cdot r}$, where m is the number of subinterval *symbols* and r is the redundancy coefficient. This quantization is performed to make watermark invisible, as will be evaluated later. Each of the mentioned subintervals represents one of the m symbols $A = \{0, 1, \ldots, m-1\}$ so that each symbol is represented by r subintervals. For the sake of simplicity we assign the symbols to subintervals in an increasing manner starting from the first subinterval, while it can be assigned in more non-uniform manners as well. To describe this mathematically, for any point of time $t \in [0, T]$ within a mark interval the symbol represented by that point is derived as:

$$sym(t) = round\left(\frac{t}{\gamma}\right) \mod m \tag{8}$$

where $round(x)$ function rounds x to the nearest smaller integer and mod performs the modular arithmetic. Figure 2 shows the assignment of symbols to subintervals in a mark interval.

For each watermark bit, the watermarker ties the secret dependant $Y(i)$ of (7) to the mark interval of that watermark bit by making specific modifications to that interval. As can be seen from Figure 1 the value of $Y(i)$ lies in the interval $[-\frac{T}{2}, \frac{T}{2}]$; so (8) can be used to find the symbol corresponding to the time value $Y(i) + \frac{T}{2}$. This symbol, $sym\left(Y(i) + \frac{T}{2}\right)$, is defined as the symbol of that mark interval, $SYM(i)$:

$$SYM(i) = sym\left(Y(i) + \frac{T}{2}\right) = round\left(\frac{Y(i) + \frac{T}{2}}{\gamma}\right) \mod m \tag{9}$$



(a) Original flow

(b) Watermarked flow

Figure 3: Delaying packets to insert watermark ($m = 3$, $r = 4$ and the symbol to insert is $SYM(i) = 1$).

We then make the $mark(i)$ interval represent the symbol $SYM(i)$ by delaying any packet within that interval to the nearest subinterval representing the symbol $SYM(i)$. The packets delayed to a subinterval are distributed uniformly at random within that subinterval, in order to avoid the accumulation of packets in certain locations, providing better watermark invisibility. Packets already in the right subinterval would be delayed to the next corresponding subinterval if they interfere with the distribution of other packets delayed into that subinterval (this is to prevent accumulation of packets in the end of subintervals representing symbol $SYM(i)$). Figure 3 shows how packets are delayed to make a mark interval represent some specific symbol. The watermark detector performs a *synchronization* process to compensate for the unknown initial offset and also network perturbations. This is discussed in Section 6.1.

It might happen that for some packet arriving at the end of a mark interval there is no subinterval representing the required symbol appearing after its arrival time. The mentioned packet is not delayed by the watermarker (e.g., last packet of Figure 3), while the same consideration is done during the detection process for these special packets, as will be described later.

As can be inferred from the described watermarking scheme, we need non-empty intervals to represent base and mark intervals; so, the watermarker and detector should ignore empty intervals during the watermarking process. We propose to ignore all the empty intervals larger than $0.75T$, as removing only empty intervals larger than T can result in destroying the watermark pattern because of the network jitter. More specifically, by removing empty intervals only larger than T network delay applied on the watermarked flow could result in changing some of the empty intervals into non-empty intervals or vice versa, e.g., by moving a packet arriving at the end of a interval into the beginning of a previously empty interval. This can ruin the synchronization of the intervals between watermarker and watermark detector, leading watermark detection to fail. To prevent this desynchronization effect by the network delay we remove all the empty intervals of length $0.75T$ or more before doing the watermarking and also during the detection process. Note that by watermark detector performing synchronization (to compensate for the effect of random initial offset), the mean of network delay does not matter in the watermarking performance, but its variance does.

2.3 Watermark detection scheme

Having access to the *watermark key*, detector is aiming to check whether some observed network flow is watermarked. The watermark key K, shared between the watermarker and the watermark detector, is the collection of secret values of watermark bits, and the corresponding base and mark intervals for each watermark bit, as in (5).

In other words, for any watermark bit i detector knows the corresponding secret value, $S(i)$, and the index of the corresponding base and mark intervals, $base(i)$ and $mark(i)$. The watermark detector checks all of the watermark bits and decides on the watermark existence if number of detected bits passes some threshold.

To check any of the watermark bits, say the i^{th} bit, we use the secret value for that bit and the centroid

of the corresponding base interval , $base(i)$, to evaluate the expected symbol for the mark interval $mark(i)$. To do so, the watermark detector uses the corresponding secret value, $S(i)$, along with the evaluated centroid of $base(i)$ interval to estimate the $Y(i)$ parameter, namely $\hat{Y}(i)$, such that

$$S(i) = D_{1,1,1}\left[(1, \hat{Y}(i)), (X_2, C(base(i)))\right] \tag{10}$$

where $C(\cdot)$ is the centroid function defined in (6). Note that as mentioned before in Section 2.1, this is equivalent to find the first-order line $f(x) = a + bx$ passing through the points $(0, S(i))$ and $(X_2, C(base(i))$ and then finding the value of this line at point $X = 1$, i.e., $\hat{Y}(i) = f(1)$. The extracted $\hat{Y}(i)$ is then used to find the expected symbol for the corresponding mark interval, $mark(i)$:

$$\widehat{SYM}(i) = sym(\hat{Y}) \tag{11}$$

As mentioned in the watermark insertion section, to do the watermarking all the packets in the mark interval $mark(i)$ are delayed into the subintervals representing the symbol of the corresponding watermark bit, $SYM(i)$. So, for any packet arriving at time t_p within the interval $mark(i)$ we check whether $sym(t_p) = \widehat{SYM}(i)$ is valid. Denote $\rho(i)$ as the ratio of such packets in $mark(i)$ interval. We call the i^{th} watermark bit to be detected, i.e., $D(i) = 1$, if this ratio passes some threshold ϱ:

$$D(i) = \begin{cases} 1 & if \quad \rho(i) \geq \varrho \\ 0 & \text{o.w.} \end{cases} \tag{12}$$

It might happen that at the insertion process the $Y(i)$ parameter is on the very close vicinity of the neighbor subinterval. This watermark bit is very susceptible to even very small amounts of network delay which moves $Y(i)$ to the neighbor subinterval, resulting to miss the corresponding watermark bit. To prevent this to happen, if detector fails to detect symbol $\widehat{SYM}(i)$ in the corresponding interval we also check for the *nearest adjacent symbol*, which is evaluated as:

$$\widehat{SYM^A}(i) = sym(\hat{Y}(i)) + sign[(Y(i) + 0.5T)/\gamma - round((Y(i) + 0.5T)/\gamma) - 0.5] \tag{13}$$

where $sign(x)$ returns the sign of its argument (see Figure 1 for the nearest adjacent symbol). The detection process for this symbol is similar to what we do for $\widehat{SYM}(i)$, and the watermark bit is declared as detected if either $\widehat{SYM}(i)$ or $\widehat{SYM^A}(i)$ symbols are detected in the candidate intervals.

Doing this for all of the watermark bits, if the total number of detected bits exceeds some detection threshold η, we declare the flow to be watermarked:

$$\begin{cases} \sum_{i=1}^{b} D(i) \geq \eta & \text{then: watermark detected} \\ \sum_{i=1}^{b} D(i) < \eta & \text{then: watermark not detected} \end{cases} \tag{14}$$

3 System analysis

3.1 False positive errors

The false positive rate is the odds of detecting a non-watermarked flow as watermarked. As mentioned before in Section 2, the detection scheme checks for two symbols for any watermark bit. Let us consider a non-watermarked flow being evaluated by DMark detector. For any packet appearing in a mark interval, e.g., interval $mark(i)$ of the non-watermarked flow, the probability of that packet representing the symbol $SYM(i)$ evaluated using (11) would be:

$$FP_{pckt} = \frac{1}{m} \tag{15}$$

This is because of the m equally likely symbols in each mark interval. For a watermark bit to be detected by the detector, we should have that at least ϱ ratio of its packets represent one of the two designated symbols. This means that for an interval with P packets, the probability of a single bit false positive would be:

$$FP_{bit}^{P} = 2 \sum_{\alpha=\lceil \varrho P \rceil}^{P} \left(\begin{array}{c} P \\ \alpha \end{array} \right) (FP_{pckt})^{\alpha} (1 - FP_{pckt})^{P-\alpha} \quad (16)$$

$$= 2 \sum_{\alpha=0}^{P-\lceil \varrho P \rceil} \left(\begin{array}{c} P \\ \alpha \end{array} \right) (1 - FP_{pckt})^{\alpha} (FP_{pckt})^{P-\alpha} \quad (17)$$

$$= 2I_{FP_{pckt}}(\lceil \varrho P \rceil, 1 + P - \lceil \varrho P \rceil) \quad (18)$$

where the last line is derived using the cumulative distribution function of the Binomial distribution, and $I.(\cdot,\cdot)$ is the regularized incomplete beta function. The evaluated FP_{bit}^{P} is the false positive probability of a single watermark bit, given that the number of packets in the considered mark interval is P. Considering the network flow as a Poisson process, the number of packets in a typical interval of length T is a Poisson distribution with the parameter λT where λ is the effective rate of the network flow. The averaged false positive for a single watermark bit would be

$$FP_{bit} = E_{P}^{\lambda T}[FP_{bit}^{P}] = \sum_{P=0}^{\infty} \frac{e^{-\lambda T}(\lambda T)^{P}}{P!} FP_{bit}^{P} \quad (19)$$

Finally, based on (14) a flow is detected as watermarked if at least η watermark bits are detected. So, the false positive rate is evaluated using the CDF of a Binomial distribution based on the false positive of a single watermark bit:

$$FP = \sum_{\alpha=\eta}^{b} \left(\begin{array}{c} b \\ \alpha \end{array} \right) (FP_{bit})^{\alpha} (1 - FP_{bit})^{b-\alpha} \quad (20)$$

$$= \sum_{\alpha=0}^{b-\eta} \left(\begin{array}{c} b \\ \alpha \end{array} \right) (1 - FP_{bit})^{\alpha} FP_{bit}^{b-\alpha} \quad (21)$$

$$= I_{FP_{bit}}(\eta, 1 + b - \eta) \quad (22)$$

where again $I.(\cdot,\cdot)$ is the regularized incomplete beta function.

3.2 False negative errors

In this section we consider false negative errors, i.e., the probability of missing a watermarked flow by the watermark detector. The probability of false negative error for a single watermark bit in a mark interval, i.e., the odds of that packet not representing the symbol corresponding to its interval, can be written as:

$$FN_{bit} = FN_{symbol} + (1 - FN_{symbol}) FN_{int} \quad (23)$$

where FN_{symbol} is the probability of error in extracting the symbol for the corresponding mark interval, and FN_{int} is the probability of missing the watermark bit given the right symbol. Based on the watermarking scheme structure, errors of smaller than γ over $Y(i)$ introduce no error to the symbol extraction of the watermark detector (since we accept two adjacent symbols during the detection). This means that the mark interval symbol is detected correctly when $C(base(i))$ deviates in an interval of length $X_2\gamma$ (see Figure 1). So, we can upperbound FN_{symbol} as:

$$FN_{symbol} = Pr(|e_C(i)| > X_2\gamma/2) \quad (24)$$

where $e_C(i)$ is the modified centroid due to noise:

$$e_C(i) = \hat{C}(base(i)) - C(base(i)) = \frac{1}{n} \sum_{j}^{n} \delta_j \quad (25)$$

where δ_j is the network delay on the j^{th} packet of that interval. We model the network delay as i.i.d. Gaussian noise, as discussed in [7]. Because of performing watermark synchronization at the watermark detector, we model network delay as zero-mean i.i.d. distributions $\delta_i \sim N(0, \sigma)$, where σ is the variance of the network delay. We get:

$$e_C(i) \sim N(0, \sigma) \tag{26}$$

so

$$\begin{aligned} FN_{symbol} &= Pr(|e_C(i)| > X_0\gamma/2) && (27)\\ &= 2\left(1 - \Phi_{0,1}\left(\frac{X_0\gamma/2}{\sigma}\right)\right) && (28)\end{aligned}$$

where $\Phi_{0,1}(\cdot)$ is the CDF function for $N(0,1)$. To evaluate FN_{int} of (23) we first find the probability of a false positive of a single packet FN_{pckt}, i.e., the probability that a single watermarked packet goes out of its subinterval because of the noise. As mentioned in Section 2.2, the watermarked packets are distributed uniformly at random within their containing intervals, in order to help with the watermark invisibility. Consider a watermarked packet in a distance x from the center of its containing subinterval ($-\gamma/2 \le x \le \gamma/2$). Again, considering the synchronization at the detector and the Gaussian distribution for the network delay we get that:

$$FN_{pckt}(x) = 1 - \Phi_{0,1}\left(\frac{\gamma/2 - x}{\sigma}\right) + \Phi_{0,1}\left(-\frac{\gamma/2 + x}{\sigma}\right) \tag{29}$$

where $FN_{pckt}(x)$ is the false negative error for the packet distanced x from the center of its containing subinterval. We derive FN_{pckt} by averaging this for different values of x, considering the mentioned uniform distribution within the subinterval:

$$FN_{pckt} = \int_{x=-\gamma/2}^{\gamma/2} FN_{pckt}(x)\frac{1}{\gamma}dx \tag{30}$$

As mentioned above, FN_{int} is the probability of missing the watermark bit given the right symbol. Considering the ϱ threshold of (12) for detecting watermark bits, we get this probability for an interval with P-packets to be

$$\begin{aligned} FN_{int}^P &= \sum_{\alpha=\lceil \varrho P\rceil}^{P} (FN_{pckt})^\alpha (1 - FN_{pckt})^{P-\alpha} && (31)\\ &= \sum_{\alpha=0}^{P-\lceil \varrho P\rceil} (1 - FN_{pckt})^\alpha (FN_{pckt})^{P-\alpha} && (32)\\ &= I_{FN_{pckt}}(\lceil \varrho P\rceil, 1 + P - \lceil \varrho P\rceil) && (33)\end{aligned}$$

The evaluated FN_{bit}^P is the false negative of a single watermark bit, given that the number of packets in the considered mark interval is P. Similar to discussions of false positive, we model network flows by a Poisson process of rate λ; so, the number of packets in a typical interval of length T is a Poisson distribution with the parameter λT. We use this to evaluate the averaged false negative of a single watermark bit:

$$FN_{int} = E_P^{\lambda T}[FN_{int}^P] = \sum_{P=0}^{\infty} \frac{e^{-\lambda T}(\lambda T)^P}{P!} FN_{int}^P \tag{34}$$

This along with (27) and (23) completes evaluation of FN_{bit}. Similar to what we did for false positive, based on (14) we find the overall false negative error:

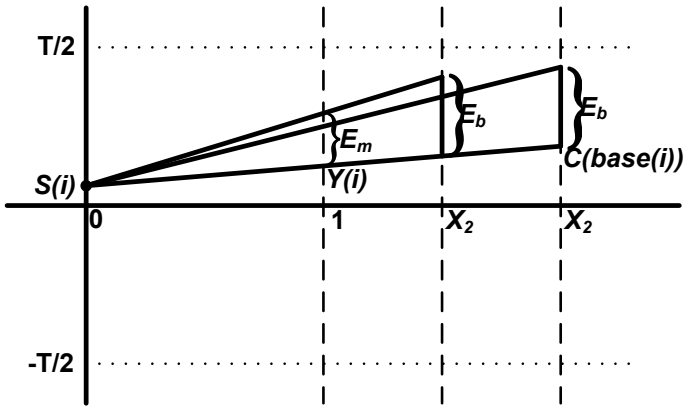


Figure 4: Selection of X_2.

$$FN = \sum_{\alpha=b-\eta+1}^{b} \left(\begin{array}{c} b \\ \alpha \end{array} \right) (FN_{bit})^{\alpha}(1-FN_{bit})^{b-\alpha} \tag{35}$$

$$= \sum_{\alpha=0}^{\eta-1} \left(\begin{array}{c} b \\ \alpha \end{array} \right) (1-FN_{bit})^{\alpha} FN_{bit}^{b-\alpha} \tag{36}$$

$$= I_{FN_{bit}}(b-\eta+1,\eta) \tag{37}$$

4 System Design

4.1 On the selection of X_2

The choice of X_2 parameter of the dependency algorithm makes a tradeoff between improving false negative error, robustness to the MFA attack, and watermark invisibility. As can be seen from Figure 4 by changing $C(base(i))$ in an interval with length E_b for a fixed value of $S(i)$ the evaluated $Y(\cdot)$ over the $X = 1$ would have a maximum deviation of:

$$E_m = \frac{1}{X_2} E_b \tag{38}$$

This shows that increasing X_2 lowers the deviation of estimated $Y(\cdot)$ caused by the errors in evaluating $C(base(i))$. On the other hand, as would be discussed in Section 7.1, for the watermark to be robust to multi-flow attack of [8] we desire that for any random selection of secret values, different symbols are equally likely assigned to the mark intervals. Suppose that the centroid of the base interval $C(base(i))$ naturally varies in an interval of length L_0 with a high probability (clearly, $L_0 < T$ and L_0 is also dependent on the flow rate). In this case, for a fixed secret value of $S(i)$, the $Y(i)$ parameter lies in an interval with length:

$$L = \frac{1}{X_2} L_0 \tag{39}$$

using the same argument as above. For all the symbols to be covered in this interval we need that:

$$L > \gamma m = \frac{T}{r} \tag{40}$$

Having this along with the constraint of (39) we get that:

Table 1: Selection of interval length parameter T with respect to the flow rate.

Effective rate (λ)	T
1-3	4000
>3	2000

$$X_2 = L_0 \frac{r}{T} \tag{41}$$

, e.g., for $r = 20$ and a conservative assumption of $L_0 = \frac{T}{5}$ we get $X_2 = 4$. Moreover, as mentioned later in (44) the delay applied by the watermarking scheme over the flows is related to $\frac{T}{r}$; so, increasing X_2 in (41) improves the watermark invisibility (by decreasing the applied delay), while the scheme is more robust to the MFA attack for smaller values of X_2 (see Section 7.1 for discussions).

4.2 Other parameters

Based on the analysis made in Section 3, we select system parameters in this part to achieve desired functionalities. In fact, playing with different system parameters makes tradeoffs between different metrics of the watermarking scheme. In other words, selection of one parameter influences the choice of other parameters in the system design.

We select $b = 32$ bits of watermark to be inserted. Watermark key of (5) is generated using some random number generator and shared between watermarker(s) and detector(s). Based on our measurements on network flows over PlanetLab the variance of the network delay is measured to be on average $\sigma = 10msec$. Choice of T depends on the effective rate of the network flow (effective rate is the rate of the flow after removing empty intervals larger than $0.75T$, as described before). In fact, as can be seen from (19) for flows with lower packet rates we need lager interval length parameter T to avoid very high false positive rates. We design DMark to use different T values for different ranges of flow rates, as shown in Table 1. In fact, using different T values is to provide faster watermark detection for higher rate flows, as the detection time is related to T (see (45)). For rare flows with effective rate of less than 1 we remove all the clear intervals of length $1000msec$ or more to assure an effective rate of more than 1 pckt/sec. Also, since a network flow might have an efficient rate falling within both of the categories, we first check for the larger rate category during the watermark insertion.

The delay applied by the watermarking scheme, which impacts watermark invisibility, is proportional to $\frac{T}{r}$ as mentioned later in Section 7.2. We set $\frac{T}{r} = 100msec$, as this makes a reasonable tradeoff between watermark invisibility, detection performance, and robustness to the MFA attack, as described later. This leads to $r = 20$ for $T = 2000$. We also choose $m = 5$ symbols resulting in $\gamma = 20msec$. Based on the discussions of Section 4.1 and having $\frac{T}{r}$ we fix the parameter of X_2 to $X_2 = 4$.

As can be seen from the analysis of Section 3 choice of packet threshold ϱ makes a tradeoff between false positive and false negative errors. In our system design, we set the packet threshold to $\varrho = 0.5$. To choose the detection threshold η we use the analysis of the false errors made before to find the optimum detection threshold value for this setting of parameters in the following.

For this set of parameters of the watermarking system, we evaluate the false positive and false negative errors evaluated in (20) and (35). Figure 5 shows these errors sketched with respect to the detection threshold η, where the effective rate of the flow is $\lambda = 4.4$pckt/sec. From the figure, false positive and false negative errors are equal for $\eta \approx 12$ pckt/sec. This false error is known as cross-over error rate (COER) which is the case of equal false negative and false positive rates.

Figure 6(a) shows the COER for different values of λ, while the corresponding detection threshold values, η_{COER}, is sketched in Figure 6(b) (η_{COER} is also known as optimum detection threshold). Detector can be set up to use a detection threshold based on the rate of the candidate flow, however we can also set a unanimous detection threshold for the watermark detection. Figure 7 shows the false positive and false negative errors for different rates of the network flow using the unanimous detection threshold of $\eta_U = 11$. As can be seen, for higher effective rates false positive error falls down very quickly.

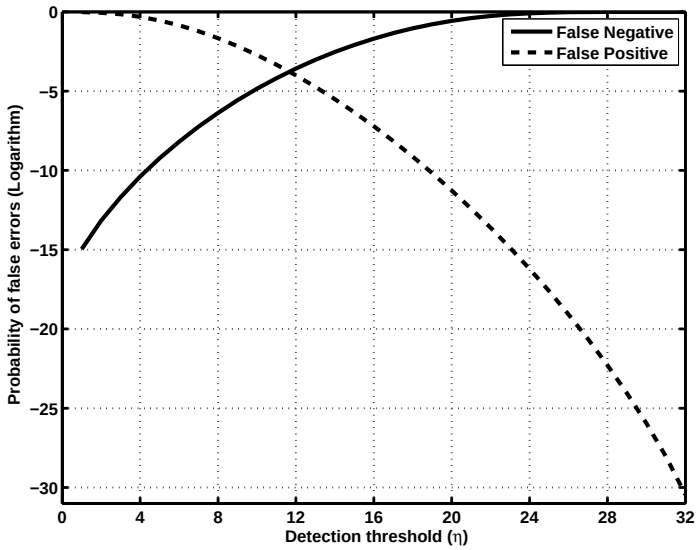
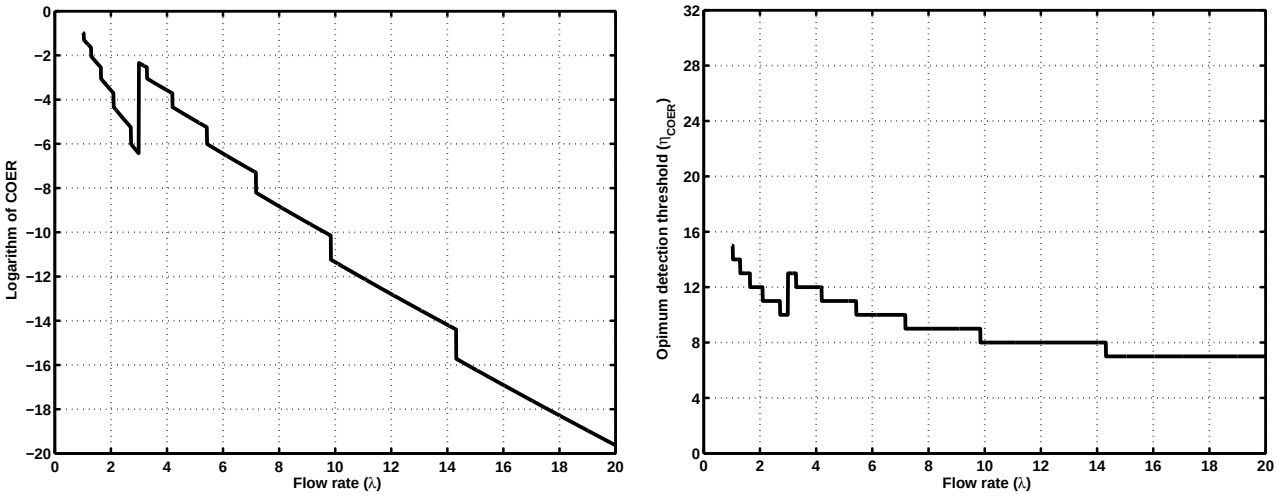


Figure 5: Analytical false positive and false negative errors for different detection thresholds ($\lambda = 4.4$ pckt/sec).



(a) COER

(b) Optimum detection threshold

Figure 6: COER and optimum detection threshold for different effective rates.

4.3 Design tradeoffs

There are different tradeoffs in the mentioned watermarking scheme which can be used to meet the required functionalities in the system design. Having interval length parameter T fixed, the choice of m and r make a tradeoff between detection efficiency and invisibility. Larger r makes the delay applied to the packets smaller, i.e., more invisible watermark, while increasing m reduces false positive error rates, as in (20).

There is also a tradeoff between speed of detection versus efficiency of the detection. As analysis show in (35) and (20) increasing the number of watermark bits b reduces false positive and false negative errors, at the expense of slower watermark detection. Interval length parameter T has also the same effect on detection efficiency and the detection time.

Moreover, as mentioned before, detection threshold η and packet threshold ϱ make tradeoffs between false positive and false negative error rates.

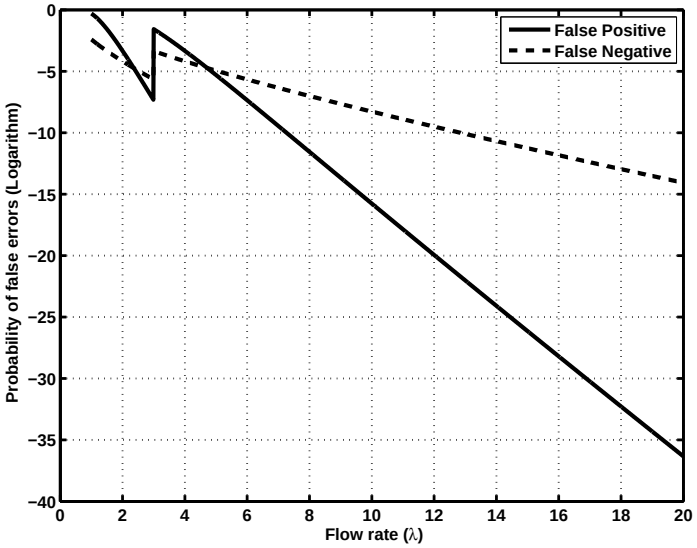


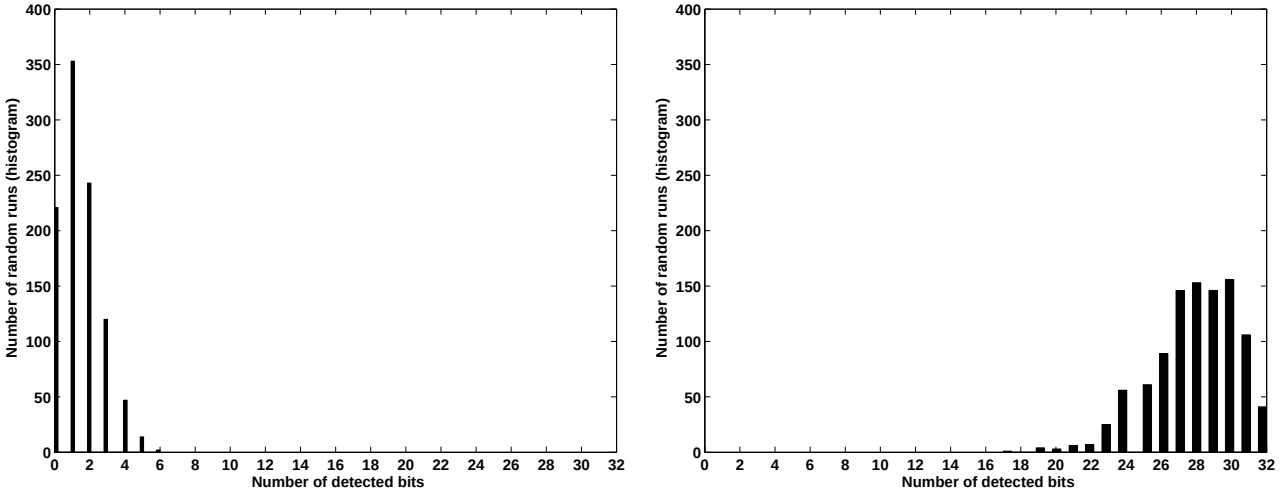
Figure 7: Probability of false positive and false negative errors for unanimous detection threshold of $\eta_U = 11$.

5 Simulations

We simulated the DMark watermarking system in Matlab. A watermark key is generated using the random number generators as specified in (5). We use $b = 32$ watermark bits, and use the system design parameters mentioned in Section 4. We use a database of network flows to represent our network flows in our simulations (the average effective rate of the flwos is 4.1 packets/sec). In every run of the simulation, a network flow is randomly selected from the database and is watermarked using the designated watermarking key. In order to consider the effect of network delay over watermarked flows we use traces of network delay measured over PlanetLab infrastructure [3]. The network delay traces are generated by capturing network delay between a number of randomly selected nodes in PlanetLab, which have different standard deviations from $\sigma = 6.20msec$ to $\sigma = 12.07msec$. For every run of the simulations a network delay sequence is selected at random and applied to the watermarked flow. Finally, the watermarked flow affected by network delay is evaluated by the simulated watermark detector to check for the known watermark. We run this experiment 1000 times, each time with the same watermark key but random selection of network flows and network delays. Figure 8(b) shows the histogram of the number of watermark bits (out of $b = 32$) detected by watermark detector out of the simulated watermarked flows, i.e., true detected bits.

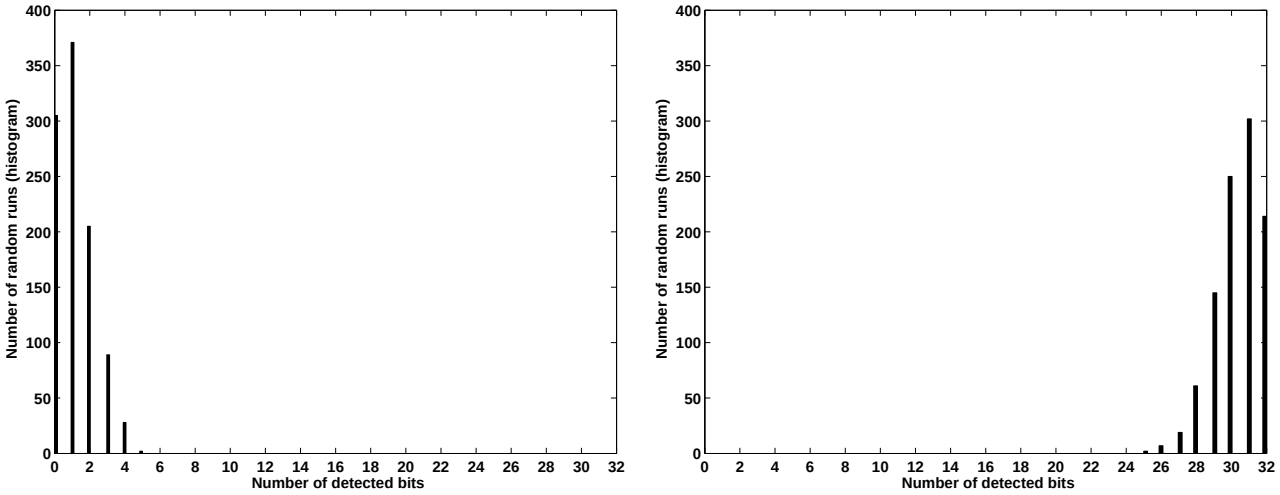
To consider false positive errors, we perform the same simulations to evaluate watermark bits detected out of non-watermarked flows. Similar to the previous experiment, we randomly select network flows from the database and apply network delay to the selected flows using the same scenario as above. We then pass the flows through the watermark detector to check for the watermarked bits. This experiment is also run for 1000 times. Figure 8(a) illustrates the histogram of the number of detected watermark bits out of the non-watermarked flows. The wide gap between the histogram of true detection and false detection bits assures the detection efficiency of th DMark, confirming the analysis made before. As can be seen, using the unanimous detection threshold of $\eta_U = 11$ no instance of false positive or false negative error occurs in the set of performed simulations.

In the mentioned experiment we had that $T = 2000$ and $r = 20$. We also run the same experiment for $T = 2000msec$ and $r = 10$. As described later in Section 7, increasing $\frac{T}{r}$ improves detection performance, by applying larger delays on the packets. Figure 9 shows the average of detected bits for this setting.



(a) False detected bits (b) True detected bits

Figure 8: Histogram of watermark bits detected by the simulated DMark detector for $\frac{T}{r} = 100$ (1000 random runs).



(a) False detected bits (b) True detected bits

Figure 9: Histogram of watermark bits detected by the simulated DMark detector for $\frac{T}{r} = 200$ (1000 random runs).

6 Implementation of DMark

We implemented the devised watermarking scheme, DMark, over the PlanetLab infrastructure to evaluate its performance. We set up system parameters similar to those of simulations, i.e., $m = 5$, $b = 32$, $\varrho = 0.5$, and T as specified in Table 1. We used different values of r to show how the T/r ratio affects the detection performance. We also generate the watermarking key randomly as described before.

We use two groups of network flows with different ranges of effective flow rate to perform the detection experiments. In the first experiment the flows are watermarked and sent over the PlanetLab to watermark detector nodes in order to evaluate watermark bits detected from the watermarked flows, i.e., *true detected bits*. In the second experiment, the non-watermarked version of same flows are passed through the detector

Table 2: Watermark detection results for the PlanetLab implementations. Analytical values E_T and E_F are provided for comparison.

Flow rate λ range (packet/sec)	r	T/r (msec)	E_T	True detected bits		E_F	False detected bits	
				Mean	Range		Mean	Range
4.28 - 4.53	10	200	31.73	30.3	27-32	2.93	5.4	4-7
	20	100	30.43	30.7	29-32	2.93	3.5	3-4
	30	66.7	27.02	27.2	15-32	2.93	2.7	2-4
5.91 - 6.20	10	200	31.93	31.6	30-32	1.35	5.0	0-6
	20	100	31.26	31.0	30-32	1.35	3.6	1-5
	30	66.7	28.55	28.1	18-32	1.35	3.6	1-5

nodes to evaluate *false detected bits*, i.e., bits detected out of non-watermarked flows. Table 2 shows the results of the PlanetLab experiments. The experiments are performed for three values of r to illustrate the effect of T/r on the system performance.

Comparing the two group of flows, the watermark detection performance improves for flows with higher rates, confirming the system analysis mentioned before. Also, within the same group of network flows decreasing r for a fixed T, i.e., increasing T/r, improves system performance by having more true detected bits and less false detected bits. Results are not so different from $r = 10$ to $r = 20$, while system performance starts to decrease more evidently for $r = 30$. As discussed in Section 7.1, selection of T/r makes a tradeoff between system performance, watermark invisibility, and robustness to multi-flow attack. The implementation results of Table 2 confirms system analysis and simulation results provided in previous sections.

In order to compare the implementation results with the analysis of Section 3 we define two paramters E_T and E_F which are true detected bits and false detected bits *estimated* using the analysis of Section 3, respectively. We use false positive error, FP_{bit}, and false negative error, FN_{bit}, of a single bit to derive these parameters:

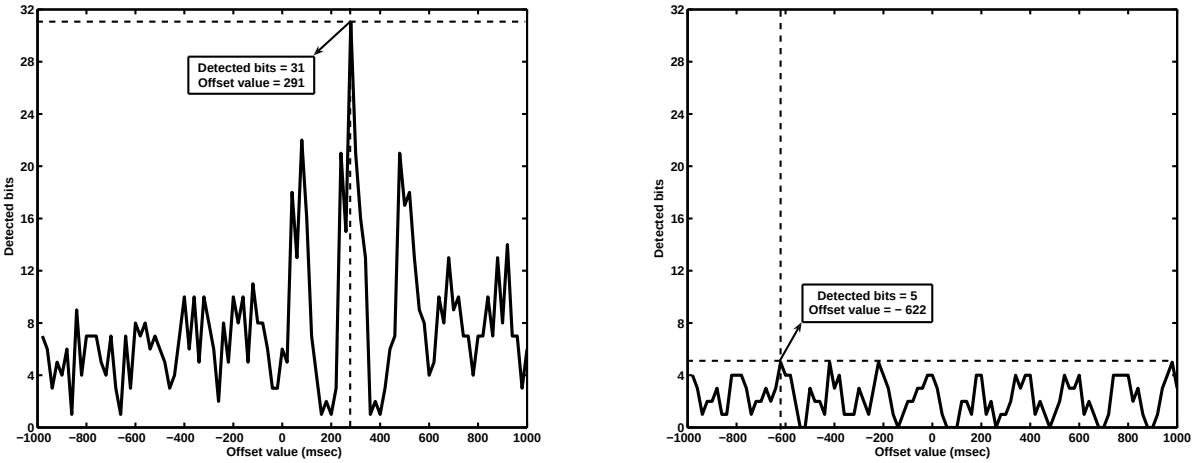
$$E_T = b(1 - FN_{bit}) \tag{42}$$

$$E_F = bFT_{bit} \tag{43}$$

where FP_{bit} and FN_{bit} are derived in (19) and (23), respectively. Table 2 compares E_T and E_F with the experimental true detected bits and false detected bits. The analysis is done considering the standard deviation of network delay in the testbed, which is measured to be $\sigma = 5msec$. As can be seen, there is a promising consistency between the analysis results and the experimental results, which indicates how good the analysis of Section 3 model the DMark watermarking system. Also, implementation results outperform the simulation results of Section 5, since the network standard deviation, σ, is twice that of PlanetLab implementation, in order to provide a conservative evaluation of the watermarking system.

6.1 Detector synchronization

The DMark watermarker applies a random initial offset in the range of $[-\frac{T}{2}, \frac{T}{2}]$ before the first watermark interval, as discribed in Section 2. In addition, network delay on the path from watermarker to detector applies additional unknown offset to the watermarked flows. In order for an efficient watermark detection, detector needs to synchronize the recieved possibly watermarked flows with the watermark intervals. We propose a simple and low-weight synchronization mechanism by trying different offsets within the range of $[-\frac{T}{2}, \frac{T}{2}]$ in $T/100$ steps, and return the maximum value as the result of the detection. Figure 10 scketchs the detected bits for different offset values using this synchronization mechanism for a watermarked flow as well as a non-watermarked flow. As can be seen, the used synchronization mechanism helps with the accurate watermark detection, while it does not ruin the false positive and false negative error rates.



(a) Watermarked flow

(b) Non-watermarked flow

Figure 10: Synchronization at watermark detection.

Table 3: Average of the number of true and false detected bits (out of 32) for different values of X_2 (averaged over 100 runs).

X_2 paramter	Mean of true detection	Mean of false detection
2	27.14	1.55
4	28.00	1.49
6	28.44	1.76
8	28.70	1.35

7 Other issues

7.1 Multi flow attack resistance

Kiyavash et al. show in [8] how multi-flow attacks (MFA) can be applied to compromise interval-based flow watermarking schemes [15, 19, 11]. The main idea of the MFA attack is to accumulate a number of network flows watermarked by an interval-based watermarking scheme using the same watermark key, and utilize the aggregation of these flows to extract watermarking parameters and the watermarking key. More specifically, attacker tries to find the common watermark patterns, e.g., empty intervals, between flows watermarked with the same watermark key utilizing the *cumulative histogram* of those flows. The attack is shown to be highly effective in compromising existing interval-based watermarking schemes of [15, 19, 11] leading to removal and/or duplication of the inserted watermark. In this section we show how MFA is inefficient against the interval-based watermarking scheme of DMark, as a benefit of doing data-dependent watermarking .

We simulated the MFA attack over the devised watermarking scheme. As mentioned in Section 4.1, selection of X_2 makes a tradeoff between detection efficiency, watermark invisibility, and robustness to MFA attack. In fact, the network delay noise over the centroid of the base interval affects the extracted $\hat{Y(\cdot)}$ parameter in the watermark detection, hence affecting the estimated symbol of the corresponding mark interval. As can be seen from Figure 4, increasing X_2 reduces the effect of centroid deviations, caused by the network delay, on the value of $Y(\cdot)$, resulting in improving detection efficiency. Table 3 depicts the mean of true detected and false detected bits for different values of X_2, reflecting the slight increase of true detected bits for larger values of X_2. On the other hand, increasing X_2 results in less variations in assigning symbols to a specific mark interval, given a fixed watermark key, which makes the watermark more threatened by the MFA attack, as is mentioned in the following.

Table 4: Variance of the symbols assigned by the watermarker for different values of X_2.

X_2 parameter	Watermark bit				Average over 32 bits
	2	6	16	30	
2	6.5000	12.5000	25.0000	12.5000	22.9063
4	22.5000	41.0000	49.0000	62.0000	43.3438
6	213.5000	145.0000	164.5000	163.0000	188.9531
8	329.5000	385.0000	293.5000	452.5000	354.4688

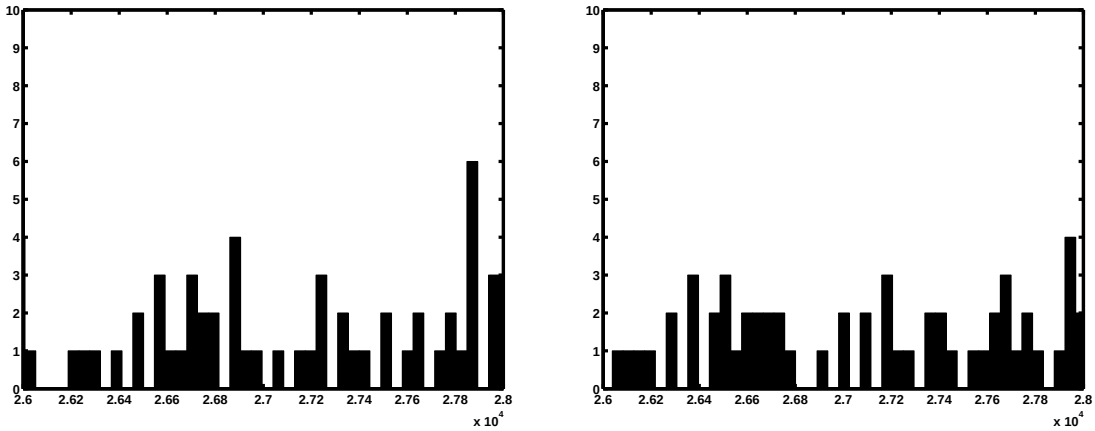
In order to be able to perform the MFA attack on our watermarking scheme, an attacker faces two issues before appllication of the attack: the random initial offset inserted by the watermarker and the empty intervals ignored in the watermarking process. Because of the initial offset appearing in the beginning of the first watermark interval, an MFA attacker needs to try different initial delays for each of captured watermarked flows in performing the attack. This not only enforces more computational complexity to attackers, but also increases the error rate of the attack. On the other hand, watermarker do not consider empty intervals with a length of more than $0.75T$, while T is unknown to the attacker. So, attacker needs to first estimate T using the mechanisms mentioned in [8] before applying the main MFA attack. Even a tiny error in estimating T corrupts synchronization of watermarked flows used to generate the cumulative histogram of MFA, making the attack infeasible.

We apply the MFA attack, assuming that attacker has successfully estimated the T parameter beforehand, which requires extra computations for the attacker. We also assume that attacker is using the corresponding initial offset for all of the captured flows (this requires an exhaustive search by the attacker for each of the received flows). Figure 11 shows the cumulative histogram of 10 watermarked flows for 2 different values of X_2 along with those of the nonwatermarked flows. To resist the MFA attack we need that cumulative histogram of watemarked flows look like that of the unwatermarked flows. As can be seen from the figure, increasing X_2 makes the watermark presence more visible. In fact, the secret behind watermark invisiblity in our watermarking scheme is to have similar packet distribution in mark and base intervals of the cumulated flows. For this to happen, we need watermarker to assign symbols to the mark intervals in an uniform manner from the set of all m possible symbols. As an example, considering a specific mark interval, if in the accumulation of 10 watermarked flows 5 symbols of 0 are assigned to that interval and no symbol of 1 is assigned to the that interval in different flows then the subintervals representing symbol 0 would contain more packets compared to subintervals of symbol 1, making it different from the distribution of packets in a base interval of the same cumulated flows (and also the histogram of non-waermarked flows). Agian, from Figure 4 making X_2 more distant from X_1 while $S(\cdot)$ is fixed results in more variations in the value of the extracted symbol. Figure 12 shows the variations of symbols assigned to a fixed watermark bit for different values of X_2 in 100 runs of the watermarking scheme using the same key. Apparently, smaller values of X_2 provides more uniform distribution of symbols in different runs of the scheme. Table 4 illustrates the variance of symbol distribution for some of the watermark bits and the average values for different values of X_2 (depicted bits are chosen randomly as a sample). As can be seen, increasing X_2 increases the variance of assigned symbols distribution, compromising watermark invisibility and making it susceptible for attacks. As mentioned before, increasing X_2 improves the false error rates (see Table 3).

In fact, even if some magical attacker finds the watermark pattern for some of the watermarked flows, i.e., knows the symbols corresponding to mark intervals for those flows, he/she is not able to reveal the watermark key, duplicate the watermark to another flow, or even check for the watermark in other flows. This is because the symbol assigned to each mark interval is dependent to some secret value and also the centroid characteristics of another unknown interval. This makes the proposed scheme even more robust to the MFA attack of [8].

7.2 Watermark invisibility

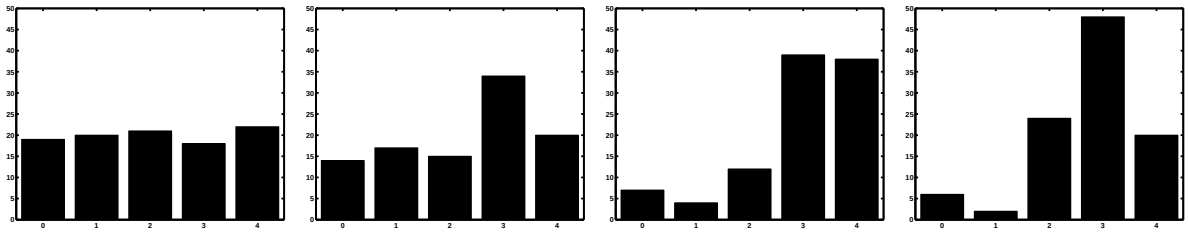
Watermark visibility is related to the amount of delay applied on the packets of the watermarked flows. Considering the symbol assignment in delaying strategy mentioned in Section 2 (see Figure 2), one can find the maximum delay applied by the DMark watermarking scheme over the packets as:



(a) $X_2 = 2$, Watermarked histogram (b) $X_2 = 2$, Non-watermarked histogram

(c) $X_2 = 8$, Watermarked histogram (d) $X_2 = 8$, Non-watermarked histogram

Figure 11: Cumulative histogram of 10 flows for two different values of X_2.



(a) $X_2 = 2$ (b) $X_2 = 4$ (c) $X_2 = 6$ (d) $X_2 = 8$

Figure 12: Distribution of symbols assigned to a sample watermark bit for different values of X_2 parameter ($m = 5$).

$$D_{max} = \gamma \times m = \frac{T}{r} \tag{44}$$

X_2 has inverse relation with $\frac{T}{r}$, as mentioned in Section 4.1; so, considering (**??**) to have less watermarking delay we need to set a larger X_2, which results in less resilience to the MFA attack based on the discussions

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

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

of the previous section. Table 5 shows the watermark delay over the packets for different values of the redundancy parameter r (T is fixed to $T = 2000msec$ and results are averaged over 500 runs). As can be seen, increasing $\frac{T}{r}$ improves detection performance at the expense of more delays on the watermarked flow packets.

Considering the tradeoffs mentioned above, the selected $X_2 = 4$ is a suitable choice as it provides reasonable detection performance, watermark invisibility, and robustness to MFA at the same time.

Another way to trade off T/r is to change T for a fixed value of r. Interval length parameter of T directly affects the total time T_{total} needed from a flow for insertion of the b-bit watermark (excluding the time of empty intervals). We have:

$$T_{total} = 2bT \tag{45}$$

which results in $T_{total} = 128sec \approx 2min$ for our setting of watermarking scheme.

7.3 Comparison with the literature

Considering detection efficiency, DMark provides error rates of less than 10^{-5} which outperforms passive traffic analysis providing errors in the order of 10^{-2} [14, 20, 18, 16, 6, 4] and blind watermarking schemes having error rates of 10^{-3} [17, 19, 15, 11]; DMark provides the same good efficiency as non-blind watermarking scheme of RAINBOW [7], while providing the advantage of blind watermark detection. As mentioned before, by doing interval-based watermarking DMark is robust to network perturbations, while also being robust to multi-flow attacks of [8] which are effective against similar interval-based schemes. Considering watermark delay, DMark is invisible by applying delays in the order of 20 msec per packet, compared to hundreds of miliseconds for similar blind watermarks [17, 19, 15, 11] and 40 msec for RAINBOW [7].

8 Scheme extension: use of higher order dependency algorithms

As discussed in Section 2.1, the $D_{1,1,1}$ dependency algorithm used in the design of DMark is motivated by Shamir's $(2, 2)$-threshold scheme. We can generally define $D_{B,M,G}$ dependency algorithms based on Shamir's $(B + 1, G + 1)$-threshold schemes, where B, M, and G represent number of base intervals, number of mark intervals, and the degree of the polynomial used, respectively. In this scenario, the number of intervals assigned to each watermark bit is $B + M$ and the total number of intervals required for the watermarking is less than or equal to $b(B + M)$, as some of the base intervals can be used for multiple watermark bits. In this section, we mention the possibility of using these extended dependency algorithms in the design of data-dependent flow watermarks and leave evaluation of their performance for the future research.

- $D_{1,n-1,1}$ algorithm: We can assign $n \geq 2$ intervals to each watermark bit by having one base interval and $n-1$ mark intervals. Similar to $D_{1,1,1}$, a linear function is used to make the dependency algorithm. Each watermark bit is declared as detected if there is a *consensus* between the results from different mark intervals. The use of this dependency algorithm helps to improve detection efficiency, at the cost of using more intervals for watermarking.
- $D_{n-1,1,n-1}$ algorithm: In this case, we use a $(n-1)$-order polynomial ($n \geq 2$) to *depend* the secret value of each watermark bit on n dependant intervals. The first $n-1$ intervals are base intervals and

the last interval is the mark interval. Use of multiple base intervals helps in more reliable detection of watermark bits.

- $D_{k-1,n-k+1,n-1}$ algorithm: This is the combination of the two previous extensions. By using this dependency algorithm we depend the watermarks on n intervals, while we use the first $k-1$ intervals as the base intervals and the remaining $n-k+1$ intervals as the mark intervals.

9 Conclusion

We proposed DMark, a novel data-dependent flow watermarking scheme. DMark uses an interval-based flow watermarking structure in order to provide robustness to network perturbations, while using dependency algorithms we make DMark robust against multi-flow attacks that are effective over similar interval-based schemes. DMark performs blind watermarking, reducing the communication overhead and computation overhead compared to passive traffic analysis and also non-blind watermarking schemes, e.g., RAINBOW [7]. We show through analysis and implementation that DMark is able to link related flows using flow lengths as long as 2 minutes, while providing error rates in order of 10^{-5} or less. Using a quantization mechanism, DMark is invisible by applying delays as low as 20 msec per packet, which is far smaller than delays of similar blind watermarking schemes, and is comparable to that of RAINBOW non-blind scheme. For the future research, we propose to explore higher order dependency algorithms in order to tradeoff different watermarking system features.

References

[1] Adam Back, Ulf Möller, and Anton Stiglic. Traffic analysis attacks and trade-offs in anonymity providing systems. In *Information Hiding*, volume 2137 of *Lecture Notes in Computer Science*, pages 245–247. Springer, 2001.

[2] Andy Bavier, Mic Bowman, Brent Chun, David Culler, Scott Karlin, Steve Muir, Larry Peterson, Timothy Roscoe, Tammo Spalink, and Mike Wawrzoniak. Operating systems support for planetary-scale network services. In Robert Morris and Stefan Savage, editors, *Symposium on Networked Systems Design and Implementation*, pages 253–266. USENIX, March 2004.

[3] Andy Bavier, Mic Bowman, Brent Chun, David Culler, Scott Karlin, Steve Muir, Larry Peterson, Timothy Roscoe, Tammo Spalink, and Mike Wawrzoniak. Operating systems support for planetary-scale network services. In Morris and Savage [2], pages 253–266.

[4] Avrim Blum, Dawn Xiaodong Song, and Shobha Venkataraman. Detection of interactive stepping stones: Algorithms and confidence bounds. In Erland Jonsson, Alfonso Valdes, and Magnus Almgren, editors, *International Symposium on Recent Advances in Intrusion Detection*, volume 3224 of *Lecture Notes in Computer Science*, pages 258–277. Springer, September 2004.

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

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

[7] Amir Houmansadr, Negar Kiyavash, and Nikita Borisov. Rainbow: A robust and invisible non-blind watermark for network flows. In *NDSS*, February 2009.

[8] Negar Kiyavash, Amir Houmansadr, and Nikita Borisov. Multi-flow attacks against network flow watermarking schemes. In Paul van Oorschot, editor, *USENIX Security Symposium*, Berkeley, CA, USA, 2008. USENIX Association.

[9] Birgit Pfitzmann and Patrick McDaniel, editors. *IEEE Symposium on Security and Privacy*, May 2007.

[10] The Tor Project. Metrics project: Measuring the Tor network. `https://www.torproject.org/projects/metrics.html.en`, 2009.

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

[12] Jean-François Raymond. Traffic analysis: protocols, attacks, design issues, and open problems. In *International Workshop on Designing Privacy Enhancing Technologies*, pages 10–29. Springer, 2001.

[13] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

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

[15] X. Wang, S. Chen, and S. Jajodia. Network flow watermarking attack on low-latency anonymous communication systems. In Pfitzmann and McDaniel [9], pages 116–130.

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

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

[18] K. Yoda and H. Etoh. Finding a connection chain for tracing intruders. In Frédéric Cuppens, Yves Deswarte, Dieter Gollmann, and Michael Waidner, editors, *European Symposium on Research in Computer Security*, volume 1895 of *Lecture Notes in Computer Science*, pages 191–205. Springer, October 2000.

[19] W. Yu, X. Fu, S. Graham, D.Xuan, and W. Zhao. DSSS-based flow marking technique for invisible traceback. In Pfitzmann and McDaniel [9], pages 18–32.

[20] Y. Zhang and V. Paxson. Detecting stepping stones. In Steven Bellovin and Greg Rose, editors, *USENIX Security Symposium*, pages 171–184, Berkeley, CA, USA, August 2000. USENIX Association.