Tracing Traffic through Intermediate Hosts that Repacketize Flows

Young June Pyun*, Young Hee Park*, Xinyuan Wang†, Douglas S. Reeves* and Peng Ning*
* Department of Computer Science
North Carolina State University, Raleigh, NC 27695, USA
Emails: {yjpyun,ypark3,reeves,pning}@ncsu.edu
† Department of Information and Software Engineering
George Mason University, Fairfax, VA 22030, USA
Email: xwangc@gmu.edu

***Abstract*—Tracing interactive traffic that traverses *stepping stones* (*i.e.*, intermediate hosts) is challenging, as the packet headers, lengths, and contents can all be changed by the stepping stones. The traffic timing has therefore been studied as a means of tracing traffic. One such technique uses traffic timing as a side channel into which a *watermark*, or identifying tag, can be embedded to aid with tracing. The effectiveness of such techniques is greatly reduced when repacketization of the traffic occurs at the stepping stones. Repacketization is a natural effect of many applications, including SSH, and therefore poses a serious challenge for traffic tracing.**

This paper presents a new method of embedding a watermark in traffic timing, for purposes of tracing the traffic in the presence of repacketization. This method uses an invariant characteristic of two traffic flows which are part of the same stepping stone chain, namely, elapsed time of the flows. The duration of each flow is sliced into short fixed-length intervals. Packet timing is adjusted to manipulate the packet count in specific intervals, for purposes of embedding the watermark. A statistical analysis of the method, with no assumptions or limitations concerning the distribution of packet times, proves the effectiveness of the method given a sufficient number of packets, despite natural and/or deliberate repacketization and perturbation of the traffic timing by an adversary. The method has been implemented and tested on a large number of synthetically-generated SSH traffic flows. The results demonstrate that 100% detection rates and less than 1% false positive rates are achievable under conditions of 2 seconds of maximum timing perturbation and 12% repacketization rate, using fewer than 1000 packets.

I. Introduction

Network-based attacks are a serious threat to the Internet community. Despite the development of sophisticated defense mechanisms, attacks continue to increase. Unfortunately, identifying the source of an attack is a challenging task due to the anonymous nature of the Internet [1] and evasion techniques [2]–[5] available to the attackers. In the case of an interactive attack, where an attacker interacts with a remote victim host using applications such as *telnet/ssh*, one of the simplest ways to hide the source of an attack is staging the attack through a series of intermediate hosts, called *stepping stones*. Tracing such an attack back to the origin is difficult since the attack is staged through several TCP connections, and the packet headers, lengths, and contents can all be changed by the stepping stones. Existing IP traceback techniques [6], [7] will only trace packets back to the last stepping stone in the sequence. Note that although this paper discusses tracing an attack, it is not confined to such traffic and can be generalized to tracing any traffic generated by interactive communication protocols (ssh/telnet). We limit our scope to interactive traffic due to its timing constraints; the amount of delays (or noise) that can be added to the traffic is bounded [8], which is essential for tracing.

This work was supported by DTO under contract NBCHC030142. The contents of this paper do not necessarily reflect the position or the policies of the U.S. Government.

Tracing interactive traffic through stepping stones requires correlating a chain of connections linked by the stepping stones. So far, techniques based on traffic timing (delays between packets in the flow) [8]–[12] have been successful in tracing such traffic. Especially, Wang and Reeves [11], [12] proposed novel methods, which use traffic timing as a side channel into which an identifying tag (called a *watermark*) can be embedded to aid with tracing traffic. They showed, through analysis and simulation, that these methods are highly robust against a limited amount of perturbation of the traffic timing (by adding delays to each packet) introduced by an adversary, in an attempt to defeat tracing.

However, the effectiveness of these watermarking techniques is greatly reduced when the traffic is transformed at the stepping stones, changing the packet count of the traffic flow. Since these tracing techniques require accurate packet synchronization between two connections being correlated, such transformations cause packet de-synchronization. Unfortunately, many such cases arise in practice, most commonly due to TCP *repacketization* [13]. Repacketization is a natural effect of many applications, including SSH [5], and therefore poses a serious challenge for traffic tracing. Moreover, since repacketization occurs when packets are buffered at the stepping stones before being forwarded, its occurrence may be increased by the adversary's timing perturbation, causing more packets to be buffered. Fig. 1 illustrates an example of packet de-synchronization caused by repacketization.

In this paper, we propose a new method of embedding a watermark in traffic timing, for purposes of tracing the traffic across stepping stones, in the presence of not only

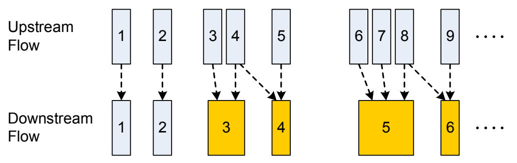


Fig. 1. Illustration of packet de-synchronization caused by repacketization. Packets $\langle 3, 4, 5, 6, 7, 8, 9 \rangle$ in the upstream flow are repacketized into packets $\langle 3, 4, 5, 6 \rangle$ in the downstream flow and are de-synchronized thereafter.

timing perturbation but also repacketization. Our correlation technique utilizes the elapsed time of the flows, which is an invariant characteristic of two connections that are part of the same stepping stone chain. The duration of each flow is sliced into short fixed-length intervals, which are used for synchronizing two connections being correlated. Packet timing is adjusted to manipulate the packet count in specific intervals, without adding or deleting any packets, for the purpose of embedding a watermark. This new method does not require precise clock synchronization between the watermark encoder and decoder, as the intervals are self-synchronized during decoding, adjusting to the adversary's timing perturbation. The analytical and experimental results show that the new method is very effective in practice, despite natural and/or deliberate repacketization and perturbation of the traffic timing, producing high detection rates and low false positive rates.

The rest of the paper is organized as follows: section 2 first reviews the related work; section 3 introduces the proposed watermarking technique with a theoretical model; section 4 analyzes the effectiveness of the method and the impact of both timing perturbation and repacketization; section 5 presents a general framework for the implementation, and section 6 evaluates the method through experiments; finally, section 7 concludes this paper with future research directions.

II. Related Work

A. Connection Correlation Techniques

Tracing interactive traffic through stepping stones requires the discovery of an association (*correlation*) between two connections at the stepping stone, such that these connections act as consecutive flows in a chain of connections. Generally, connection characteristics which remain unchanged (*i.e.*, invariant) at the intermediate host are used to determine whether two connections are correlated. The traffic can be then traced back to the origin by linking the correlated connections together. So far, substantial research has been done on such connection correlation techniques.

Following the early work based on packet payloads [3], [14], techniques based on traffic timing were proposed to deal with encrypted connections [15], [16]. In these techniques, the timing characteristic of an interactive connection was assumed to be unique and preserved across stepping stones such that it can be utilized for correlating connections [16].

Faced with the potential for an adversary to use active timing perturbation at the stepping stones in an attempt to defeat timing-based correlation, more sophisticated timing-based techniques were proposed later. They analyzed and compared the long-term behavior [8] and packet counts [10] of the connections, assuming that the adversary's delays are bounded.

As opposed to the above passive approaches, Wang and Reeves [11], [12] proposed active timing-based correlation techniques that are robust against random timing perturbation. Their method embeds unique watermarks into connection flows, by actively manipulating packet timing of the flows, such that these watermarks can be identified when correlating connections. These active approaches have shown to be very successful for traffic tracing.

A number of new techniques were also introduced to address dummy packets (called *chaff*) that can be added into traffic by an adversary, in addition to the timing perturbation. These techniques provide lower bounds on the amount of chaff required to evade detection [10] and upper bounds on the number of packets needed for a confident detection [17], and compare tradeoffs among detection rates, false positive rate, and computation cost [18].

B. Watermarking for Connection Correlation

Traditionally, *digital watermarking* has been developed as a technical means for protection of intellectual property, such as copyright protection and distribution tracing. Watermarking generally provides methods to embed (*encode*) information (*watermark*) transparently into a carrier signal, such that it can be retrieved (*decoded*) from the signal later. In addition to preserving the carrier signal quality, it is required to be robust against countermeasures intended to distort or remove the embedded information from the signal. Without the knowledge of secret parameters (keys), as in cryptography, the knowledge of its existence alone should not make it easy to remove. Detailed information on watermarking can be found in [19].

In recent years, Wang and Reeves [11], [12] have adopted conventional watermarking concepts to the domain of traffic tracing. Specifically, their methods actively encode a unique watermark w into an upstream flow of the stepping stone connection chain by manipulating the *inter-packet delays* (IPDs) of certain pre-selected packet pairs, such that these changes are propagated to the downstream flows. If w is robust enough, it is highly likely to be recognized in the downstream flows but nowhere else. Here, the packet timing is considered as a carrier signal, and changes in the packet timing are considered as the watermark. Since the decoder does not have the original traffic at hand, but only the watermarked traffic, these techniques can be classified as *semiblind watermarking* [19].

The two versions of the above techniques mainly differ in the process of embedding the watermark, *i.e.*, how the packet pair selection and their timing adjustment are accomplished. In order to encode a single watermark bit, the earlier method [11] uses a single IPD, delaying the second packet by an amount that results in an IPD that is a multiple of a specified step size. The binary watermark bit is encoded/decoded by the quantization level. This technique is comparable to conventional *Quantization Index Modulation* [20], which is based on quantizing the host signal vector to transmit a message.

The second method [12] is more efficient and more robust, but is also more complex. In this method, the second order timing difference (*i.e.*, the difference between the IPDs of two packet pairs) is used, and it is the *average* second order timing difference of two groups of packet pairs into which the watermark value is encoded. This is accomplished either by increasing the IPDs of the first group of packet pairs, and/or by decreasing the IPDs of the second group of packet pairs. Encoding a watermark bit with value 0 requires that the average IPD of the first group be greater than the average IPD of the second group, while encoding a value of 1 requires the opposite. This technique is similar to *Patchwork* watermarking method [21], which modifies a statistic of two components of a signal in opposite directions in order to encode a value.

III. Interval-based Watermarking

This paper proposes an innovative and practical watermarking technique for purposes of tracing traffic. It extends the previous watermarking methods [11], [12], so as to provide robustness against not only timing perturbation but also repacketization—a critical issue in the previous work.

The proposed technique uses the elapsed time of the flows, which is an invariant characteristic of two flows that are part of the same stepping stone chain. The duration of each flow is partitioned into short fixed-length intervals which are used for synchronization purposes between the two flows. Packet timing is then adjusted to manipulate the packet count of the intervals, in an attempt to encode a watermark into the upstream flow, so that it can be unambiguously decoded in the downstream flow. Here, the watermark is no longer tied to any specific packet. It is rather bound to the intervals, which are unaffected by changes in packet count of the flows. This technique is comparable to conventional *block-based patchwork techniques* [22] used for still images, where a sequence of bits are modified rather than just one bit, for a higher resistance to lossy data compression.

A. Watermarking Model

Let a flow F be selected from a large pool of interactive traffic flows. The selected flow F with a stream of packets $\langle P_1, P_2, \ldots\rangle$, indexed with respect to the arrival timing order, is divided into intervals I_i ($i > 0$, denoting the interval index) of time T (>0) starting from a random time offset o (>0). An interval I_i then contains X_i contiguous packets, where X_i is a random variable representing the number of packets in the interval I_i. Due to the random offset associated with each different flow F, an interval I_i is located randomly with respect to the remaining flows in the pool. Hence, it follows that $X_1, X_2, \ldots$ are identically distributed with respect to all the flows. Here, we make no assumption on the distribution of X_i but will denote the mean as μ_x and the variance as σ_x^2, whose values are both proportional to the interval length T.

Then, r (>0) pairs of 2 consecutive intervals are randomly selected. For each pair of the intervals, the first interval is denoted as $I_{1,k}$, whereas its counterpart is denoted as $I_{2,k}$ ($k = 1, \ldots, r$). Assuming a long lasting flow consisting of many intervals, $X_{1,k}$ ($k = 1, \ldots, r$) are random samples from a sufficiently large population (X_i, $i > 0$) of a common distribution and hence independent among themselves. Hence, $X_{1,k}$ ($k = 1, \ldots, r$) are *iid*—this is true for $X_{2,k}$ ($k = 1, \ldots, r$) as well. Consequently,

$$E(X_{1,k}) = E(X_{2,k}) = \mu_x, \quad V(X_{1,k}) = V(X_{2,k}) = \sigma_x^2.$$

However, each of the pairs ($X_{1,k}$ and $X_{2,k}$) may not be independent of each other, depending on the distribution of X_i. In fact, since interactive traffic exhibits ON-OFF activity with bursts of packets and inter-arrival times of the packets in Pareto distribution [9], [23], we assume $Corr(X_{1,k}, X_{2,k}) \geqslant 0$.[1]

Let the packet count difference of the above selected interval pair be

$$Y_k = \frac{X_{1,k} - X_{2,k}}{2} \quad (k = 1, \ldots, r). \tag{1}$$

Then, $E(Y_k)$=0 and $V(Y_k) \leqslant \sigma_x^2/2$, where the maximum variance occurs when $Corr(X_{1,k}, X_{2,k}) = 0$. Furthermore, since $X_{1,k}$ and $X_{2,k}$ are *iid* of their own, so are Y_k ($k = 1, \ldots, r$).

Now with the sample size of r, representing a *redundancy*, the sample mean of Y_k is defined as

$$\overline{Y_r} = \frac{1}{r}\sum_{k=1}^{r} Y_k\,. \tag{2}$$

Then, since Y_k ($k = 1, \ldots, r$) are *iid*, it can be easily shown that $E(\overline{Y_r})=0$ and $V(\overline{Y_r}) \leqslant \sigma_x^2/2r$. Note that, the distribution of Y_k is symmetric about the mean ($=0$) since Y_k represents the difference of two random variables that are of the same distribution. Hence, the distribution of $\overline{Y_r}$ is also symmetric about the mean with its variance decreasing as we increase r.

B. Watermark Encoding/Decoding

As in the conventional watermarking and cryptography, the watermarking method relies on a shared secret (*key*) between the *encoder* and the *decoder*. The new method assumes that the following parameters are pre-distributed: a random offset $o > 0$, an interval length $T > 0$, an embedding interval selection function S, and a binary watermark w of l bits. Once the watermark embedding interval pairs are selected, the encoder *encodes* the intended watermark w onto a given flow F. Similarly, given a watermarked flow F^w, the decoder *decodes* a watermark w' and then *compares* it against w for a correlation result.

Specifically, the interval-based watermarking encodes a watermark bit by either increasing or decreasing $\overline{Y_r}$ by an amount of μ_x, depending on the given value of the watermark bit. According to (1) and (2), increasing $\overline{Y_r}$ by μ_x can be achieved by increasing $\frac{1}{r}\sum_{k=1}^{r} X_{1,k}$ by μ_x and simultaneously decreasing $\frac{1}{r}\sum_{k=1}^{r} X_{2,k}$ by μ_x. The former is accomplished by

[1]Our experiment confirmed that the synthetic SSH traffic flows, generated with tcplib [24], show X_i and X_{i+1} ($i > 0$) being positively correlated.

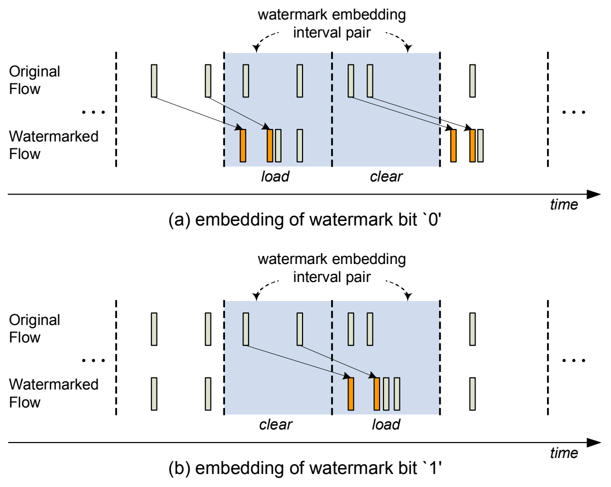


Fig. 2. Illustration of single watermark bit encoding: bit '0' and bit '1'.

loading each of the corresponding intervals $I_{1,k}$ $(k=1,\ldots,r)$; all the packets in the preceding interval are shifted into this interval by adding a maximum delay $d=T$ (called *timing adjustment*) to each of the packets.[2] The latter is accomplished by simply *clearing* each of the corresponding intervals $I_{2,k}$ $(k=1,\ldots,r)$; all the packets are delayed to the following interval in the same manner. Note that, in order to avoid conflicts between embedding interval pairs, at least one non-embedding interval is required as a buffer between any two successive embedding interval pairs. Decreasing $\overline{Y_r}$ by μ_x can be done in the opposite way.

To encode a watermark bit '0', $\overline{Y_r}$ is increased as stated above so that the result, denoted as $\overline{Y_r^w}$, is positive ($\overline{Y_r^w} > 0$). On the other hand, to encode a watermark bit '1', $\overline{Y_r}$ is decreased so that the result is negative ($\overline{Y_r^w} < 0$). It not guaranteed that $\overline{Y_r}$ can be increased; the preceding interval may happen not to contain any packets. The use of redundant coding reduces the likelihood that a particular value cannot be coded because of an absence of packets. In addition, the watermark itself is l bits in length, and is tolerant to small errors in coding. Fig. 2 only illustrates the encoding of a single bit, with no redundancy $(r=1)$.

The encoded watermark bit can then be easily decoded. Let $\overline{Y_r^{w'}}$ at a decoder be a random variable comparable to $\overline{Y_r^w}$ at an encoder. To decode a watermark w' of a watermarked flow F^w, $\overline{Y_r^{w'}}$ is computed over the same embedding intervals used for encoding w and then compared against 0; a positive result implies a watermark bit of '0', whereas a negative result implies a watermark bit of '1'. A correct correlation is achieved if $w=w'$.

IV. Theoretical Analysis

A. Probabilistic Watermark Success Rate

In the interval-based watermarking, there is a non-zero probability such that an encoded watermark bit cannot be decoded correctly. That is, if a watermark resulted in $\overline{Y_r^w} \leqslant 0$ for $w=0$ ($\overline{Y_r^w} \geqslant 0$ for $w=1$) for any reason, the decoded watermark w' may not coincide with the intended watermark w and may hence fail to correlate the flows correctly. In order to measure the performance of the interval-based watermarking, we define the probability that a watermark bit is decoded correctly as a *watermark success rate* and express it as $Pr\left[\overline{Y_r^w} > 0\right]$ for $w=0$ ($Pr\left[\overline{Y_r^w} < 0\right]$ for $w=1$). Due to the symmetry of the distribution of $\overline{Y_r}$, only the former case ($w=0$) is analyzed in this paper.

First, since $Y_1,\ldots,Y_r$ are random samples, so are $Y_1^w,\ldots,Y_r^w$ after a watermark is encoded. Hence, assuming a large sample size of r, the distribution of $\overline{Y_r^w}$ can be approximated to a standard normal distribution, according to the *Central Limit Theorem* [25];

$$Pr\left[\frac{\left(\overline{Y_r^w} - E(\overline{Y_r^w})\right)}{\sqrt{V(\overline{Y_r^w})}} > \xi\right] \approx 1 - \Phi(\xi), \tag{3}$$

where

$$\Phi(\xi) = \int_{-\infty}^{\xi} \frac{1}{\sqrt{2\pi}} e^{-u^2/2} du\,.$$

Let $X_{1,k}^w$ and $X_{2,k}^w$ denote the respective results of $X_{1,k}$ and $X_{2,k}$ after a watermark bit '0' is encoded. Then, $X_{1,k}^w = X_{1,k} + X_{0,k}$ and $X_{2,k}^w = 0$ $(k=1,\ldots,r)$. Since $E(X_i) = \mu_x$ and $V(X_i) = \sigma_x^2$ $(i>0)$, we have

$$\begin{aligned} E(X_{1,k}^w) &= 2\mu_x \\ E(X_{2,k}^w) &= 0 \quad (k=1,\ldots,r) \end{aligned}$$

while

$$\begin{aligned} V(X_{1,k}^w) &= \sigma_x^2 + \sigma_x^2 + 2Corr(X_{1,k}, X_{0,k})\sigma_x^2 \leqslant 4\sigma_x^2 \\ V(X_{2,k}^w) &= 0 \quad (k=1,\ldots,r), \end{aligned}$$

where the maximum variance occurs when $Corr(X_{1,k}, X_{0,k}) = 1$. Then, it follows that

$$E(\overline{Y_r^w}) = \mu_x, \quad V(\overline{Y_r^w}) \leqslant \sigma_x^2/r.$$

Now, when applied to (3), the watermark success rate can be approximated as follows:

$$\begin{aligned} Pr\left[\overline{Y_r^w} > 0\right] &= Pr\left[\frac{\left(\overline{Y_r^w} - E(\overline{Y_r^w})\right)}{\sqrt{V(\overline{Y_r^w})}} > \frac{-E(\overline{Y_r^w})}{\sqrt{V(\overline{Y_r^w})}}\right] \\ &\approx 1 - \Phi\left(\frac{-E(\overline{Y_r^w})}{\sqrt{V(\overline{Y_r^w})}}\right) \\ &\geqslant 1 - \Phi\left(\frac{-\sqrt{r}\mu_x}{\sigma_x}\right). \end{aligned} \tag{4}$$

This derivation implies that, for a given r, the watermark success rate is decreased as μ_x decreases and/or σ_x^2 increases, and vice versa. On the other hand, however, this rate can always be increased to as high as 1, regardless of the exact distribution of X_i as long as $\mu_x > 0$, by increasing the redundancy r. Note that, this result accounts for the worst case, where $Corr(X_i, X_{i+1}) = 1$.

[2]Although there may be an upper bound on the increase of an X_i due to the limited transmission rate, its impact is usually negligible since it is highly likely that its counterpart can be decreased in such a case.

B. Impact of Repacketization and Timing Perturbation

It is plausible that an adversary may actively manipulate his traffic that might carry a watermark, in an attempt to remove or distort the watermark. One can easily delay the packets of the flow at the stepping stone, which would transform the traffic timing. This timing perturbation is challenging since it is, in most cases, accompanied by repacketization of the flow, which transforms the packet count of the flow as well as its timing. We now analyze the negative impact of these transformations on the watermark success rate of the interval-based watermarking.

First, the proposed method assumes the delay model of Donoho *et al.* [8] where an adversary is confined to conservative transformation due to the inherent constraint of the interactive traffic; i) there is a limit on the maximum delay an adversary can tolerate when perturbing his traffic, and ii) these delays are short-termed local jittering and cannot transform the long-term characteristic of the traffic [8]. Furthermore, these delays are, regardless of their distribution, assumed to be random with respect to the watermark embedding packets since the attackers have no knowledge of where these watermarks are embedded. Additionally, the amount of repacketization on interactive traffic is assumed to be bounded as well, due to the maximum transmission unit (MTU) of the underlying network and availability of the data generated by the user; most interactive traffic consists of slow key stroke commands followed by small result data and a long OFF-periods.

Let $X^p_{1,k}$, $X^p_{2,k}$, and $\overline{Y^p_r}$ denote the respective results of $X^w_{1,k}$, $X^w_{2,k}$, and $\overline{Y^w_r}$ after the watermarked flow F^w $(w=0)$ is randomly perturbed and/or repacketized $(k=1,\ldots,r)$. Now let $D \geqslant 0$ be the maximum delay an adversary is able to add to each packet. In addition, let $P_{1,k}$ be a random variable representing a decrease of the first packet count $X^w_{1,k}$ as a result of timing perturbation and/or repacketization; its mean and variance are denoted as μ_p and σ_p^2, respectively. Similarly, let $Q_{2,k}$ $(0 \leqslant Q_{2,k} \leqslant P_{1,k} \leqslant X^w_{1,k})$ be a random variable representing an increase of the second packet count $X^w_{2,k}$ due to the same cause; its mean and variance are denoted as μ_q and σ_q^2, respectively. Note that $P_{1,k}$ and $Q_{2,k}$ are both *iid* of their own as with $X^w_{1,k}$ and $X^w_{2,k}$, although they may be dependent to each other. In fact, both are proportional to X_i and D, and conversely proportional to T to a certain extent.

Then,

$$\begin{aligned} X^p_{1,k} &= X^w_{1,k} - P_{1,k} \\ X^p_{2,k} &= X^w_{2,k} + Q_{2,k} \quad (k=1,\ldots,r). \end{aligned}$$

It follows that,

$$\begin{aligned} E(X^p_{1,k}) &= 2\mu_x - \mu_p \\ E(X^p_{2,k}) &= \mu_q \quad (k=1,\ldots,r) \end{aligned}$$

while

$$\begin{aligned} V(X^p_{1,k}) &\leqslant 4\sigma_x^2 + \sigma_p^2 \\ V(X^p_{2,k}) &= \sigma_q^2 \quad (k=1,\ldots,r), \end{aligned}$$

where the maximum variances occur when $Corr(X^w_{1,k}, P_{1,k}) = 0$ $(Corr(X^w_{1,k}, P_{1,k}) \geqslant 0)$. As a result,

$$E(\overline{Y^p_r}) = \mu_x - \frac{\mu_p + \mu_q}{2}, \quad V(\overline{Y^p_r}) \leqslant (\sigma_x^2 + \frac{\sigma_p^2 + \sigma_q^2}{4})/r;$$

the maximum variance occurs when $Corr(X^p_{1,k}, X^p_{2,k}) = 0$ $(Corr(X^p_{1,k}, X^p_{2,k}) \geqslant 0)$. Finally, when applied to the *Central Limit Theorem* in (3), the watermark success rate, taking account of both timing perturbation and repacketization, is approximated to

$$\begin{aligned} Pr\left[\overline{Y^p_r} > 0\right] &\approx 1 - \Phi\left(\frac{-E(\overline{Y^p_r})}{\sqrt{V(\overline{Y^p_r})}}\right) \\ &\geqslant 1 - \Phi\left(\frac{-\sqrt{r}(\mu_x - \frac{1}{2}(\mu_p + \mu_q))}{\sqrt{\sigma_x^2 + \frac{1}{4}(\sigma_p^2 + \sigma_q^2)}}\right). \end{aligned} \tag{5}$$

When compared to the previous result (4) without any transformation, the lower bound of the success rate is decreased. As a result of both timing perturbation and repacketization, the packet count of a loaded interval has reduced, whereas that of a cleared interval has increased, reducing the overall mean of the packet count difference $(E(\overline{Y^w_r}) \geqslant E(\overline{Y^p_r}))$. Moreover, the increase in the variance of packet count of the embedding interval pairs has increased the overall variance of the packet count difference.

The embedded watermark may be completely removed if $2\mu_x = \mu_p + \mu_q$. This occurs when the maximum delay D of the adversary is considerably long, compared to the interval length T, such that a large portion of the watermark embedding packets are shifted out of their original interval into another. Hence, an interval length T should be as large as the average timing perturbation (about $D/2$) in order to retain as many packets as possible in their original embedding intervals. However, as long as $2\mu_x > \mu_p + \mu_q$, the watermark success rate can be increased with a higher redundancy, despite shorter interval length $(T < D/2)$.

V. Implementation

A. Watermarking Parameters

The choice of watermarking parameters $\langle o, T, S, w\rangle$, in addition to their secrecy, is crucial to watermarking when dealing with sophisticated attackers trying to defeat the method through techniques such as timing analysis [26]. In fact, the interval-based watermarking technique can be configured in various ways depending on the following performance goals: 1) *robustness* against active/passive transformations of the watermark, which may interfere with correct correlation, 2) *efficiency* in terms of number of required packets for a successful correlation, and 3) *stealthiness* of the watermark parameters, protecting them against timing analysis of an adversary. Therefore, these parameters should be configured in a way to balance the tradeoffs among the above three performance goals. A general strategy in the choice of the parameters is as follows:

- A random offset $o > 0$ is used for *robustness*, but its value should be reasonably small for *efficiency*.
- A unique watermark string w of length $l > 0$ should be long enough for *robustness*, avoiding any false correlation, but short enough for *efficiency*.
- An interval length $T > 0$ should be long enough for *robustness* and short enough for *stealthiness* since packet timing is adjusted with a maximum delay of $d = T$. For increased stealthiness and robustness, the interval length can be randomized for each different watermark bit.
- An interval selection function S should be able to provide randomness in its selection of the watermark embedding interval pairs for both *robustness* and *stealthiness*. Additionally, it should be able to minimize the total number of required packets without affecting the other performance factors.

For an example, to minimize the number of required packets, S can select every second and third intervals as the embedding interval pairs, starting from an offset o. For an l-bit watermark without a redundancy ($r = 1$), this requires a total of $3l$ intervals and each watermark bit does not interfere with others. Now the redundancy can be implemented by repeating the whole watermark string one after another. This particular scheme is very effective since the redundancy, hence the confidence in the result, can be increased dynamically as a watermark is encoded/decoded iteratively, adapting to the availability of the packets of a given flow. To further improve stealthiness, the watermark bit sequence is randomized at each iteration. With the random bit ordering and the random interval length, it would be difficult for an adversary to infer the parameters, even if S is known to him.

B. *Watermark Encoding/Decoding Procedures*

Having the parameters $\langle o, T, S, w \rangle$, specified as suggested above, an encoding procedure is quite simple. Let intervals $I_{i,j,1}$ denote non-embedding intervals preceding embedding interval pairs denoted as $I_{i,j,2}$ and $I_{i,j,3}$ for a jth bit of an l-bit watermark (denoted as w_j) in an ith repetition of a total redundancy r. Then, a watermark w is encoded into an upstream flow F through a procedure as follows.

Watermark encoding procedure:

1) For each incoming packet P_k of F, identify its interval index ($\geqslant 1$) using the interval selection function S, the interval length T, and the offset o.
2) Delay P_k by an amount of T if:
   - P_k belongs to $I_{i,j,1}$ and w_j indicates a ‘0’.
   - P_k belongs to $I_{i,j,2}$ and w_j indicates a ‘1’.
   - P_k belongs to $I_{i,j,3}$ and w_j indicates a ‘0’.

   If P_k is to be delayed into a new interval, past the interval boundary, the intended delay should be reduced by an random amount, such that the reduced delay still places the packet in the new interval.
3) Otherwise, add a minimal delay (*e.g.*, $5ms.$) to P_k, if its inter-packet delay with P_{k-1} is to be to small.

In step 2), the delays are randomized with a maximum value of T to increase the stealthiness of the watermark parameters. In addition, the last step helps to avoid unintended repacketization of two packets, whose inter-packet delay became very close ($< 5ms.$) as a result of the timing adjustment of the first packet.

Now, a decoder with the same parameters used to encode w is able to decode a watermark w' from a downstream flow F' and match it against w. Let $x_{i,j,2}$ and $x_{i,j,3}$ be packet counts in $I_{i,j,2}$ and $I_{i,j,3}$, respectively. In addition, let o' be a decoder's offset and w'_i be a decoded watermark after ith repetition of r, with $w'_{i,j}$ denoting its jth bit. Furthermore, let $H(w', w)$ represent a Hamming distance between w' and w. The decoding procedure is as follows.

Watermark decoding procedure:

1) Set the initial decoder's offset o' to $o - |\delta|$, where $|\delta|$ indicates a maximum clock discrepancy between an encoder and a decoder. Then, $(o - 2|\delta|) \leqslant o' \leqslant o$.
2) For each incoming packet P_k of F', identify its interval index ($\geqslant 1$) using the interval selection function S, the interval length T, and the offset o'.
3) For each of the embedding interval pairs $I_{i,j,2}$ and $I_{i,j,3}$, compute $x_{i,j,2}$ and $x_{i,j,3}$, respectively.
4) After completion of each ith repetition of r for every watermark bits, compute $X_{i,j,2} = i^{-1} \sum_{u=1}^{i} x_{u,j,2}$ and $X_{i,j,3} = i^{-1} \sum_{u=1}^{i} x_{u,j,3}$, and then compute a watermark w'_i of the ith redundancy:
   - $w'_{i,j}$ is set to ‘0’ whenever $X_{i,j,2} > X_{i,j,3}$.
   - $w'_{i,j}$ is set to ‘1’ whenever $X_{i,j,2} < X_{i,j,3}$.
   - Otherwise, $w'_{i,j}$ is set to $w'_{i-1,j}$ for $i > 1$ ($w'_{1,j} =$‘0’).

   Compare w'_i with w and report a positive detection of the watermark w if the Hamming distance $H(w'_i, w) \leqslant h$ ($0 \leqslant h < l$), where h is a decoding threshold. Continue until rth repetition or the end of the flow.
5) In the case of a negative result at the end of step 4), repeat from step 2), incrementing o' by v ($0 < v < T$).

The last step is required for synchronizing the intervals. This *self-synchronization* process not only obviates precise clock synchronization between the watermark encoder and decoder, but also is able to self-synchronize to an offset that decodes the closest matching watermark, even in the presence of random timing perturbation, shifting the intervals along with the delayed packets. In a real-time system, each trial with different offsets can be performed in parallel.

A decoding threshold h is used to filter out any noise in the process. Although a higher threshold increases the *detection rate* (the possibility of correctly identifying a connection flow as being watermarked with w), it also increases the *false positive rate* (the possibility of falsely identifying a connection flow as being watermarked with w). It is suggested that h be chosen to balance the tradeoff between the detection rate and the false positive rate [11], [12].

VI. Experiments

We implemented a real-time watermarking system to empirically evaluate the effectiveness of the new watermarking technique on a real network. The proposed method was implemented on Linux systems following the guidelines described in section V-A. The observation and manipulation of the packet streams were handled using *netfilter iptables* [27].

In order to create the effect of transformations at the stepping stones, a small network of 3 hosts was set up as shown in Fig. 3. The $Encoder/Decoder$ implemented the encoding/decoding procedures described in section V-B. The $Perturber$ simulated a set of stepping stones that may transform flows through timing perturbation and repacketization. Note that, the repacketization occurred as per the standard network protocols (*tcp* and *ssh*), whereas the timing perturbation was generated by a user application.

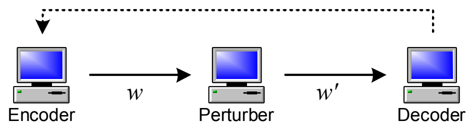


Fig. 3. Illustration of the experimental network set-up.

In each experiment, the $Decoder$ generated a series of synthetic SSH interactive flows, using *tcplib* [24], connecting to the $Encoder$. We used 50 random flows, each with at least 2000 packets and an average packet rate of 0.86 $packets/second$. However, these interactive traffic flows exhibited the ON-OFF periods as shown in [9], [23]; 75% of the packets had inter-arrival times less than $500ms$. Hence, the effective packet rate, defined as the total number of packets over the packet burst time, was much higher than the average.

For each of the incoming flows, the $Encoder$ then encoded a randomly generated 24-bit watermark w onto the reverse flow, destined back to the $Decoder$ through the $Perturber$. In the experiments with timing perturbation, the watermarked flow was perturbed with random delays before being repacketized by the $Perturber$. Finally, the $Decoder$ decoded a watermark w' from the incoming flow, which is then compared against w for a match.

A detection rate was measured, at each round of repetitions during the decoding, as the number of watermarked flows that are correctly identified as embedding the encoded watermark, over the total of 50 flows. Similarly, a false positive rate was measured, for a given watermark, as the number of unwatermarked flows that are falsely identified as embedding the given watermark, over the 50 flows.

A. Detection Rate under an Ideal Condition

In the first experiment, the watermark detection rate was evaluated when there was no timing perturbation, although repacketization may still occur. This experiment used an interval length of $T = 900ms$, which achieved a redundancy up to $r = 20$, using an average of 1200 packets. Decoding thresholds (h) ranging from $h = 3$ to $h = 7$ were used when decoding

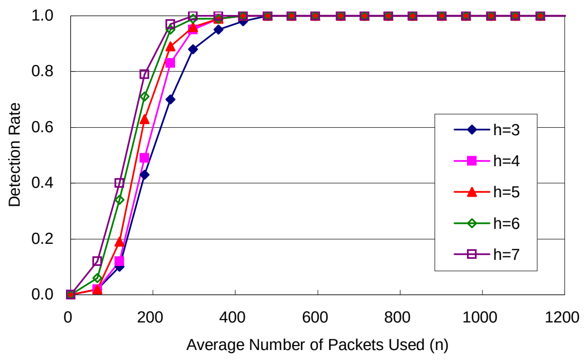


Fig. 4. Detection rate comparison of different decoding thresholds (h) as a function of the number of packets used (n). $T = 900ms$ and $D = 0ms$.

the watermarks. In this particular experiment, without any timing perturbation, repacketization occurred with at most 0.3% reduction of the total number of packets.

Fig. 4 presents the detection rate as a function of the average number of packets used (denoted as n). The result shows that the detection rate increases, regardless of the given threshold h, as the redundancy r (hence the packet usage n) increases, which is consistent with the analysis in section IV. It also shows that the new watermarking technique is very effective, even in the presence of repacketization (0.3%). It requires no more than 400 packets for a 100% detection rate in all cases of h. Although a higher decoding threshold achieved a higher detection rate for a given number of packets, it may also increase the false positive rate. A separate experiment (result not shown) demonstrated that false positive rates less than 0.5% could only be achieved when $h \leqslant 6$, in this ideal condition.

Note that, this result is a significant improvement over the existing two techniques ([11], [12]), which failed completely when faced with even this degree of repacketization. Due to the packet de-synchronization, both methods produced detection rates lower than 1%, regardless of the amount of packets they used.

B. Impact of Repacketization and Timing Perturbation

The following experiments evaluated the detection rate when faced with active timing perturbation accompanied by passive repacketization of the watermarked flow. The timing perturbation was modeled using uniformly random delays with a maximum value of D ranging from 0 to $2000ms$. Although longer delays could be introduced by the adversary, we believe that a maximum of 2 seconds is the realistic limit an adversary is able to tolerate in an interactive attack.

First, the correlation between the repacketization and the timing perturbation was tested. For each different maximum delays D, the repacketized packet ratio (defined as a *repacketization rate*) was measured. The result in Fig. 5 demonstrates that the repacketization occurs more frequently, as the delays become more significant. The repacketization rate increases to as high as 12% as D increases to $2000ms$. However, this trend levels off at about 10% (or $D = 1000ms$) due to the

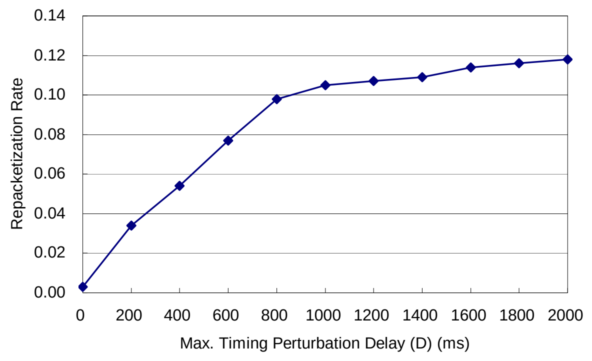
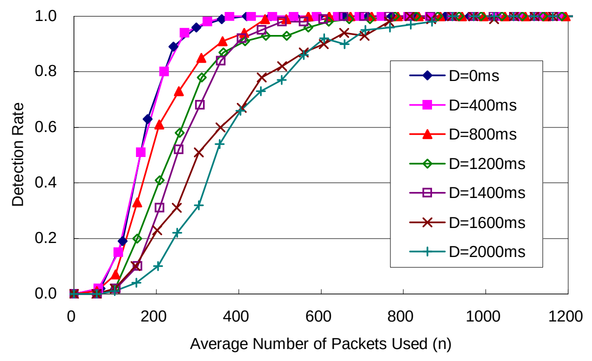
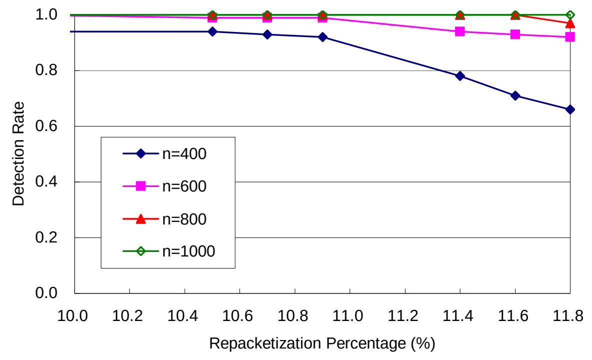


Fig. 5. Correlation between repacketization and timing perturbation. Repacketized packet ratio over a total of 5000 packets.



(a) Various degrees of timing perturbation. $T = 900ms$ and $h = 5$.



(b) Detection rate vs. repacketization rate. $T = 900ms$ and $h = 5$.

Fig. 6. Detection rate under timing perturbation and repacketization.

MTU and/or the given packet arrival rate explained in section IV-B; the effective packet rate of the sample flows used in the experiments was less than 4 $packets/sec.$, excluding the long OFF periods.

Then, the effect of different degrees of timing perturbation was evaluated, along with the accompanied repacketization. Since the timing perturbation delayed the overall packet timing of the flow, the synchronization process (described in section V-B) was used at the decoder to shift the interval boundaries along with the delayed packets. This can be achieved in parallel using multiple *iptable* rules with different decoding offsets (o'). For simplicity, offsets were added up with 60%–80% of the maximum delay D. In this experiment, we used an interval length $T = 900ms$ and a decoding threshold $h = 5$.

Fig. 6(a) shows the detection rates for each different D. It shows that, for the same number of packets used, the detection rate decreases as D increases. Similarly, for a 100% detection rate, the required number of packets are doubled in the worst case. As stated in section IV-B, this occurs as a result of timing perturbation, which shifts the watermarked packets out of their embedding intervals, and repacketization, which reduces the number of watermarked packets. As a result, the packet counts of the loaded intervals are decreased, while those of the cleared intervals are increased, decreasing their differences. Nevertheless, detection rates higher than 90% are achieved despite a maximum delay of 2 seconds, using only a few hundreds of packets. In all cases, the respective false positive rates measured no more than 1% for $h \leqslant 5$ (results not shown).

To evaluate the effectiveness of the proposed method in relation to the repacketization, the above detection rate result is combined with the result shown in Fig. 5. Fig. 6(b) presents the new result, each with different number of packets used (n). It shows that, for $n = 600$, a 100% detection rate is maintained up to a repacketization rate of 10.9% ($D = 1400ms$) before it starts to drop (though still $> 90\%$) at higher degrees of repacketization. However, even in such cases, the detection rate can be increased using a higher redundancy, *i.e.*, more packets ($n > 600$). Note that this result accounts for the repacketization caused by timing delays of low data rate traffic (0.86 $packets/sec.$), which is more challenging than

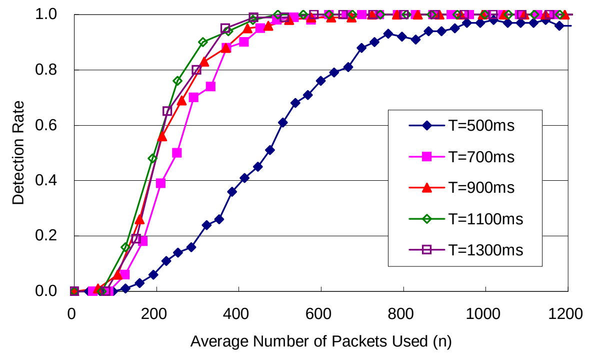


Fig. 7. Detection rate comparison of different interval lengths (T). $D = 1000ms$ (10.5% repacketization) and $h = 5$.

repacketization of higher rate traffic. In the latter case, where the repacketization occurs more frequently with even small delays (less packet shifting across intervals), our technique is able to deal with more aggressive repacketization rates, as shown in the analysis (5). In fact, for a fixed maximum timing perturbation, this method is more effective on traffic with a higher data rate.

To further investigate the effect of different interval lengths with respect to the timing perturbation, the detection rate was evaluated using different interval lengths T, ranging from $500ms$ to $1300ms$, for a fixed timing perturbation of $D = 1000ms$ (10.5% repacketization). The result in Fig. 7 demonstrates that the detection rate is proportional to T.

Especially, the performance gain becomes significant when the interval length T is small (*e.g.*, $T = 500ms$), which validates the analysis. On the other hand, when T is not so much smaller than D, then the benefit of using larger interval length becomes minimal, which suggests using a smaller T for the sake of stealthiness.

VII. Conclusion and Future Direction

Interactive traffic that traverses stepping stones is commonly subject to repacketization at the stepping stones, which changes the packet counts of the traffic flows as well as the timing characteristics. Any connection correlation technique that relies on accurate packet synchronization may hence fail due to packet de-synchronization. In this paper, we presented a new approach of correlating stepping stone connections so as to trace them back to their origin, even in the presence of timing perturbation and repacketization. This approach is based on actively manipulating the packet timing of an upstream flow for the purpose of embedding a watermark that is to be identified in the downstream flows. It utilizes time intervals for synchronization and exploits packet counts of the intervals for embedding the watermark.

The paper presented a theoretical model for the proposed watermarking technique and analyzed its effectiveness. It also identified performance tradeoffs among robustness, efficiency, and stealthiness, and introduced a general framework for implementing the method based on these tradeoffs. The real-time experiments using synthetically-generated SSH traffic flows demonstrated that the method is highly effective, producing 100% detection rates and less than 1% false positive rates using fewer than 1000 packets, despite 2 seconds of maximum timing perturbation and 12% repacketization at the stepping stones.

Although this new technique provides robustness against packet count transformation, there are other challenges that have not been addressed in this paper. Our future work will address transformations, such as adding chaff packets into a flow or, splitting/merging of flows. We will also investigate the robustness and stealthiness of our method against potential timing analysis. In addition to these countermeasures, we also need to work on scalability issues for a practical deployment of the proposed method.

References

[1] W. T. Strayer, C. E. Jones, I. Castineyra, J. B. Levin, and R. R. Hain, "An Integrated Architecture for Attack Attribution," BBN Technologies, 10 Moulton Street, Cambridge, MA 02138, Tech. Rep. BBN REPORT-8384, Dec. 2003.

[2] L. T. Heberlein and M. Bishop, "Attack Class: Address Spoofing," in *Proc. of the 19th National Information Systems Security Conference (NISSC)*, Oct. 1996, pp. 371–377.

[3] S. Staniford-Chen and L. T. Heberlein, "Holding Intruders Accountable on the Internet," in *Proc. of the 1995 IEEE Symposium on Security and Privacy (S&P)*, May 1995, pp. 39–49.

[4] D. Goldschlag, M. Reed, and P. Syverson, "Onion Routing for Anonymous and Private Internet Connections," *Communications of the ACM*, vol. 42, no. 3, pp. 39–41, Feb. 1999.

[5] T. Ylonen, "The Secure Shell (SSH) Protocol Architecture," IETF RFC:4251, Jan. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4251.txt

[6] S. Savage, D. Wetherall, A. R. Karlin, and T. Anderson, "Practical Network Support for IP Traceback," in *Proc. of ACM SIGCOMM*, Sep. 2000, pp. 295–306.

[7] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, B. Schwartz, S. T. Kent, and W. T. Strayer, "Hash-Based IP Traceback," in *Proc. of ACM SIGCOMM*, Sep. 2001, pp. 3–14.

[8] D. L. Donoho, A. G. Flesia, U. Shankar, V. Paxson, J. Coit, and S. Staniford, "Multiscale Stepping-Stone Detection: Detecting Pairs of Jittered Interactive Streams by Exploiting Maximum Tolerable Delay," in *Proc. of the 5th International Symposium on Recent Advances in Intrusion Detection (RAID)*, Oct. 2002, pp. 17–35.

[9] Y. Zhang and V. Paxson, "Detecting Stepping Stones," in *Proc. of the 9th USENIX Security Symposium*, Aug. 2000, pp. 171–184.

[10] A. Blum, D. X. Song, and S. Venkataraman, "Detection of Interactive Stepping Stones: Algorithms and Confidence Bounds," in *Proc. of the 7th International Symposium on Recent Advances in Intrusion Detection (RAID)*, Oct. 2004, pp. 258–277.

[11] X. Wang and D. S. Reeves, "Robust Correlation of Encrypted Attack Traffic through Stepping Stones by Manipulation of Interpacket Delays," in *Proc. of the 10th ACM conference on Computer and Communications Security (CCS)*, Oct. 2003, pp. 20–29.

[12] X. Wang, S. Chen, and S. Jajodia, "Tracking Anonymous Peer-to-Peer VoIP Calls on the Internet," in *Proc. of the 12th ACM conference on Computer and Communications Security (CCS)*, Nov. 2005, pp. 81–91.

[13] J. Postel, "Transmission Control Protocol," IETF RFC:793, Sep. 1981. [Online]. Available: http://www.ietf.org/rfc/rfc793.txt

[14] X. Wang, D. S. Reeves, S. F. Wu, and J. Yuill, "Sleepy Watermark Tracing: An Active Network-Based Intrusion Response Framework," in *Proc. of the 16th International Conference on Information Security (IFIP/Sec)*, Jun. 2001, pp. 369–384.

[15] K. Yoda and H. Etoh, "Finding a Connection Chain for Tracing Intruders," in *Proc. of the 6th European Symposium on Research in Computer Security (ESORICS)*, Oct. 2000, pp. 191–205.

[16] X. Wang, D. S. Reeves, and S. F. Wu, "Inter-Packet Delay Based Correlation for Tracing Encrypted Connections through Stepping Stones," in *Proc. of the 7th European Symposium on Research in Computer Security (ESORICS)*, Oct. 2002, pp. 244–263.

[17] L. Zhang, A. Persaud, A. Johnson, and Y. Guan, "Stepping Stone Attack Attribution in Non-Cooperative IP Networks," Iowa State University, Tech. Rep. TR-2005-02-1, Feb. 2005.

[18] P. Peng, P. Ning, D. S. Reeve, and X. Wang, "Active Timing-Based Correlation of Perturbed Traffic Flows with Chaff Packets," in *Proc. of the 2nd International Workshop on Security in Distributed Computing Systems (SDCS)*, Jun. 2005, pp. 107–113.

[19] M. Arnold, M. Schmucker, and S. D. Wolthusen, *Techniques and Applications of Digital Watermarking and Content Protection*, ser. Computer Security. Boston: Artech House, 2003.

[20] B. Chen and G. W. Wornell, "Quantization Index Modulation: A Class of Provably Good Methods for Digital Watermarking and Information Embedding," *IEEE Transaction on Information Theory*, vol. 47, no. 4, pp. 1423–1443, May 2001.

[21] W. Bender, D. Gruhl, N. Morimoto, and A. Lu, "Techniques for Data Hiding," *IBM Systems Journal*, vol. 35, no. 3 & 4, pp. 313–336, 1996.

[22] G. Langelaar, J. van der Lubbe, and J. Biemond, "Copy Protection for Multimedia Data based on Labeling Techniques," in *Proc. of the 17th Symposium on Information Theory in the Benelux*, Enschede, The Netherlands, May 1996, pp. 33–39.

[23] V. Paxson and S. Floyd, "Wide-Area Traffic: The Failure of Poisson Modeling," *IEEE/ACM Transactions on Networking*, vol. 3, no. 3, pp. 226–244, Jun. 1995.

[24] P. B. Danzing and S. Jamin, "tcplib: A Library of TCP Internetwork Traffic Characteristic," Computer Science Department, University of Southern California, Report CS-SYS-91-01, 1991.

[25] D. D. Wackerly, W. Mendenhall III, and R. L. Scheaffer, *Mathematical Statistic with Application*, 6th ed. Duxbury, 2002.

[26] P. Peng, P. Ning, and D. S. Reeves, "On the Secrecy of Timing-Based Active Watermarking Trace-Back Techniques," in *Proc. of the 2006 IEEE Symposium on Security and Privacy (S&P)*, May 2006, pp. 334–349.

[27] Netfilter iptables. [Online]. Available: http://www.netfilter.org