Invisible Flow Watermarks for Channels with Dependent Substitution, Deletion, and Bursty Insertion Errors (Draft)

Xun Gong, *Student Member, IEEE*, Mavis Rodrigues, Negar Kiyavash, *Member, IEEE*

Abstract

Flow watermarks efficiently link packet flows in a network in order to thwart various attacks such as stepping stones. We study the problem of designing good flow watermarks. Earlier flow watermarking schemes mostly considered substitution errors, neglecting the effects of packet insertions and deletions that commonly happen within a network. More recent schemes consider packet deletions but often at the expense of the watermark visibility. We present an invisible flow watermarking scheme capable of enduring a large number of packet losses and insertions. To maintain invisibility, our scheme uses quantization index modulation (QIM) to embed the watermark into inter-packet delays, as opposed to time intervals including many packets. As the watermark is injected within individual packets, packet losses and insertions may lead to watermark desynchronization and substitution errors. To address this issue, we add a layer of error-correction coding to our scheme. Experimental results on both synthetic and real network traces demonstrate that our scheme is robust to network jitter, packet drops and splits, while remaining invisible to an attacker.

This work was supported in part by AFOSR under Grant FA9550-11-1-0016, MURI under AFOSR Grant FA9550-10-1-0573, and NSF CCF 10-54937 CAR. This work was presented in part at ICASSP'12 [1].

X. Gong is with the Coordinated Science Laboratory and the Department of Electrical Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA (email: xungong1@illinois.edu)

M. Rodrigues was with the Coordinated Science Laboratory and the Department of Electrical Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA (email:mavis.rodrigues@gmail.com)

N. Kiyavash is with the Coordinated Science Laboratory and the Department of Industrial and Enterprise Systems Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA (email: kiyavash@illinois.edu)

I. Introduction

Detecting correlated network flows, also known as flow linking, is a technique for traffic analysis with wide applications in network security and privacy. For instance, it may be utilized to expose a stepping stone attacker who hides behind proxy hosts. Alternatively, flow linking has been successfully used to attack low-latency anonymity networks such as Tor [2], where anonymity is compromised once end flows are correctly matched. As network connections are often encrypted, it is infeasible to link flows directly relying on packet contents. However, matching flows using side information such as packet timings is possible, as their values remain to some extent unchanged even after encryption [3]–[5].

Earlier work in flow linking was based on long observation of flow characteristics, such as the number of ON/OF periods [4]. Such *passive* techniques are fragile vis-a-vis network artifacts and require long observation periods to avoid large false alarm rates. *Flow watermarking*, an active approach, was suggested as an improvement. In this approach, a pattern, the watermark, is injected into the flow with the hope that the flow stays traceable after traversing the network as long as the same pattern can be later extracted [2], [6]–[10]. Unlike passive schemes, flow watermarking is highly reliable and works effectively on short flows.

The challenge of designing good flow watermarks is to keep the injected pattern robust to network artifacts yet invisible to watermark attackers.[1] The robustness requirement guarantees that the injected pattern survives network artifacts, while the invisibility property prevents watermark removal attempts by active attackers. Most state-of-the-art schemes currently trade off one of the two properties at the expense of the other. In the so called *interval-based* schemes [2], [8], a flow is divided into intervals, and all packets within selected intervals are shifted to form a watermark pattern. Given that a few packets would not greatly affect the pattern created in the entire interval, these schemes are robust against network artifacts such as packet drops and splits. However, shifting a large number of packets produces noticeable 'traces' of the embedded watermarks and compromises the invisibility requirement [11]. In *inter-packet-delay (IPD)-based* schemes [6], [9], the delays between consecutive packets are modulated to embed watermarks. Since only small perturbations are introduced in each inter-arrival time, watermarks are not visible. The drawback of this approach is that any packet loss or insertion during transmission can cause

[1]The goal of watermark attackers is to prevent the success of flow linking by disrupting the detection or altogether removing the watermarks from the flow.

watermark desynchronization and severe decoding errors.

In this paper, we present a new IPD-based flow watermarking scheme where invisible watermark patterns are injected in the inter-arrival-time of successive packets. We treat the network as a channel with substitution, deletion, and bursty insertion errors caused by jitter, packet drops, and packet splits or retransmission, respectively, and introduce an insertion, deletion and substitution (IDS) error-correction coding scheme to communicate the watermark reliably over the channel. At the same time, we preserve watermark invisibility by making unnoticeable modifications to packet timings using the QIM framework [12]. Through experiments on both synthetic and real network traces, we show that our scheme performs reliably in presence of network jitter, packet losses and insertions. Furthermore, we verify the watermark invisibility using *Kolmogorov-Smirnov* [13] and *multi-flow-attack* tests [11]. Deletion correction codes were first applied to flow watermarking in [1], where watermarks can be decoded correctly after packet losses as long as the first packet in the flow was not dropped. In this work, we extend our decoder to handle more realistic network environments where not only packet losses but also packet insertions occur. Furthermore, synchronization requirement on the first packet is relaxed. To verify the performance of our scheme, traffic traces collected from real SSH connections are tested. This improves the simulation setup in [1], where merely simulated synthetic traffic was used.

The rest of the paper is organized as follows. Background on flow watermarking appears in §II. We describe notations and definitions in §III. Our proposed scheme is presented in §IV. We evaluate the performance of our scheme using synthetic and real traffic traces in §V.

II. Background

This section covers some background material on flow watermarking. First, we describe three application scenarios of flow watermarking. Second, we discuss some principles for designing good watermarking schemes. We conclude by surveying the literature.

A. Applications

We begin with a stepping-stone detection scenario where flow watermarks are used to find hidden network attackers. Figure 1 depicts an attacker *Bob* who wants to attack a victim *Alice* without exposing his identity. *Bob* first remotely logins to a compromised intermediate host *Charlie* via SSH [14]. Then he proceeds by sending attack flows to *Alice* from *Charlie*'s machine. Tracing packet flows sent to Alice's

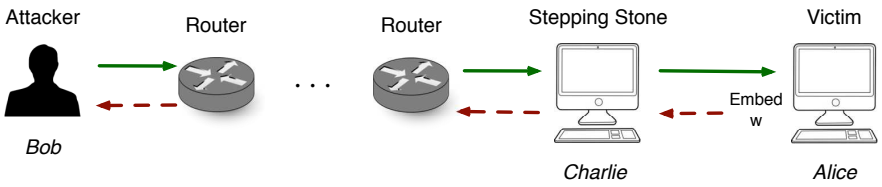


Fig. 1. Detecting attackers behind stepping stones. *Bob* uses *Charlie* as a stepping stone to attack *Alice* so that his identity remains hidden from *Alice*. To traceback the origin of this attack, *Alice* injects a watermark on the flow sent back to the stepping stone. The path leading to *Bob* is exposed as every router along this path detects *Alice*'s watermark on flows passing through.

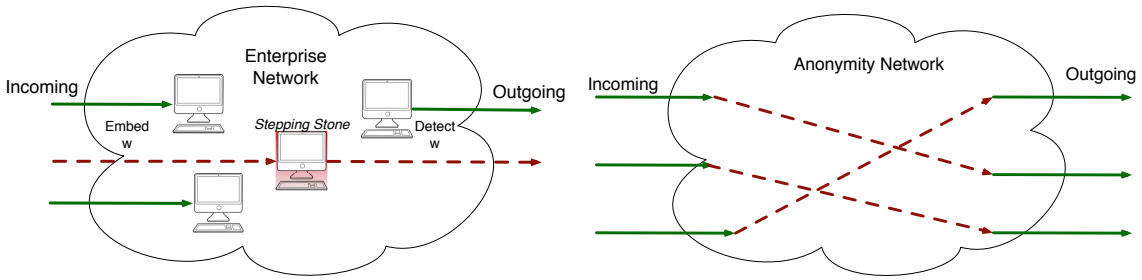


Fig. 2. (a) Stepping stones in enterprise networks. An intruder compromises a host in the enterprise network as a 'stepping stone'. The enterprise embeds watermarks on all incoming flows and monitors all the outgoing flows. Any pair of incoming/outgoing flows with the same watermark indicates the existence of inside stepping stones. (b) An anonymity network. Incoming flows are shuffled before leaving the system to hide the pairing among communicating parties.

machine would implicate *Charlie* instead of *Bob* as the attacker. Hosts like *Charlie*, exploited to hide the real attack source, are called as *stepping stones* [3]. In real life, attackers may hide behind a chain of stepping stones, making it hard for the victim, who only sees the last hop, to determine the origin of the attack. Fortunately, flow watermarking is a solution for tracing the attack source. Notice that an interactive connection is maintained along *Bob-Charlie-Alice* during the above stepping stone attack. Hence *Alice* can secretly embed a watermark in the packet flow heading back to *Charlie*. As this flow travels back to *Bob*, the watermark could be subsequently detected by the intermediate routers (or firewalls), revealing the attack path and its true origin [15], [16].

Another scenario of stepping-stone attack occurs in enterprise networks, as shown in Figure 2(a). Here, intruders are trying to compromise hosts in an enterprise network to relay their malicious traffic [11], [17]. To discover this kind of 'stepping stones' within the network, an enterprise can add watermarks on all incoming flows, and then terminate outgoing flows that contain the watermark since they most probably come from stepping stones. In a similar fashion, flow watermarking may be applied to attacking anonymity

network systems [14], [18]–[20]. In order to hide the identities of communicating parties, an anonymity network shuffles all the flows passing through it, as shown in Figure 2(b). If an attacker somehow discovers the hidden mappings between incoming and outgoing flows, the anonymity is compromised. Akin to the previous enterprise network scenario, this can be achieved by marking all incoming flows with watermarks and subsequently detecting the watermarks on the exiting flows.

B. Design Principles

From above application examples, we summarize a list of principles for designing flow watermarks. The challenge of building an efficient scheme lies in the difficulty of achieving all desired properties simultaneously.

- *Robustness*. One major advantage of flow watermarking over passive traffic analysis is the robustness against network noise. Take the stepping stone attack of Figure 1 for example. The flow *Alice* sends back to *Bob* is subjected to jitter, packet drops, and packet splits during transmission. All these artifacts can alter the watermark, resulting in decoding errors. Without the ability to withstand these artifacts, flow watermarking is no different than passive analysis, which is fragile by nature.
- *Invisibility*. A successful watermark pattern should stay 'invisible' to avoid possible attacks. For instance, in Figure 2(a), if the intruder notices that incoming flows contain watermarks, it can command the stepping stone to take precautionary actions (for instance, remove the watermarks altogether).
- *Blindness*. In a blind watermarking scheme, the watermark pattern can be extracted without the help of the original flow [21]. On the contrary, the original flow must be present in order to detect non-blind watermarks. Again, consider the example of Figure 2(a). In order to detect the stepping stone, the enterprise needs to perform watermark decoding on all outgoing flows. If a non-blind detection scheme is used, all exit routers are required to obtain a copy of each incoming flow. The resulting overheads of bandwidth and storage make such schemes impractical in large enterprise networks.
- *Presense watermarking*. In conventional digital watermarking (e.g., multimedia watermarking), often a large amount of hiding capacity is desired as the injected watermarks are frequently used to achieve copyright among many users [22]. This, fortunately, is not required for most flow watermarking applications, since the main purpose of injecting watermarks here is to link flows initiated from the same sources. In other words, in digital watermarking terminology, *zero-bit* or *presence* watermarks

suffice [21]. Therefore, when designing a flow watermarking scheme, one may trade the capacity for other properties such as robustness (see the discussion in §IV-B).

C. Watermark Attack Models

The difficulty of maintaining watermark invisibility depends on the specific attack model. Based on the strength of the watermark attacker, attack models may be classified as follows:

- *Level-I:* the attacker observes the watermarked flow, and has knowledge of certain feature (e.g., empirical distributions of IPDs) of the original flow;
- *Level-II:* the attacker observes the watermarked flow, and has a distorted version of the original flow;
- *Level-III:* the attacker observes both the watermarked flow and the original flow.

In Level-I, the weakest attack model, the attacker can only discover the presence of watermark by statistical approaches that real a deviation of know features from the norm in the original flow. For interval based schemes, the multi-flow attack (MFA) exposes empty intervals in the combination of several watermarked flows [11]. For IPD-based schemes, the empirical distribution of IPDs, which should not be changed with high probability, distinguishes watermarked flows from unwatermarked ones via Kolmogorov-Simirnov (K-S) tests [9], [13]. We show in §V-C that our watermark does not introduce noticeable patterns for the MFA or the KS test to detect it.

In Level-II, given a distorted version of the original flow, the attacker has in effect an imperfect realization of the original flow signal which is more informative than the statistical information a Level-I attacker has. A Level-II attack, BACKLIT was recently proposed, where the attacker serves as a traffic relay between the client and server of a TCP connection and thus sees both REQUEST and RESPONSE flows [24]. When watermarks are added, packets along one direction (i.e., RESPONSE) must be delayed. The attacker can detect this 'delayed' timing pattern as he observes the 'clean' flow in the REQUEST direction. BACKLIT works well when a strong correlation between the REQUEST and RESPONSE flows exists, in which case the attack has a high fidelity version of the original flow. In §V-C, we evaluate our scheme against BACKLIT. We show that in practice the correlation between the response and request flows are destroyed for the most part by network jitter because watermarks in our scheme that add very small perturbations to IPDs can remain hidden.

A Level-III attacker, who observes the exact original flow has a significantly easier task detecting the presence of a watermark [23]. This attack model, however, requires the attacker to be able to observe

arbitrary flows everywhere in the network, which is impossible for most real applications. In this work, we focus on the first two attack models when evaluating watermark invisibility.

D. Related Work

We briefly review previous flow watermarking literature. To the best of our knowledge, all the previous schemes fail to meet at least one of the above design principles, necessitating the development of a comprehensive approach that meets all the aforementioned criteria.

Earlier flow watermarks are of *inter packet delay (IPD)-based* type. In [6], the authors propose an IPD-based scheme that modulates the mean of selected IPDs using the QIM framework. Watermark synchronization is lost if enough packets are dropped or split. Therefore the scheme is unreliable. Another IPD-based scheme is presented in [9], where watermarks are added by enlarging or shrinking the IPDs. This non-blind scheme achieves some watermark resynchronizations when packets are dropped or split, but is not scalable as the original packet flow is required during decoding.

In *interval-based* schemes, instead of using the IPDs between individual packets, the watermark pattern is encoded into batch packet characteristics within fixed time intervals. In [2], an interval-centroid scheme is proposed. After dividing the flow into time intervals of the same length, the authors create two patterns by manipulating the centroid of packets within each interval. The modified centroids are not easily changed even after packets are delayed, lost or split. A similar design is presented in [8], where the watermark pattern is embedded in the packet densities of predefined time intervals. One problem with interval-based schemes is the lack of invisibility. Moving packets in batches generates visible artifacts, which can expose the watermark positions. Based on this observation, a multi-flow attack (MFA) was proposed in [11]. The authors showed that by lining up as few as 10 watermarked flows, an attacker can observe a number of large gaps between packets (see Figure.10 in [11]) in the aggregate flow, revealing the watermark positions. Recently, a new interval-based scheme was proposed in [28]. The main idea is that the exact locations of modified intervals depends on the flow pattern. This flow-dependent design reduces the success rate of MFA, but makes it more difficult to retrieve the correct intervals for decoding in face of strong network noise. Moreover, the perturbation introduced in the IPDs is large enough to make the scheme susceptible to Level-II attacks such as BACKLIT.

Table I compares existing flow watermarking schemes with our proposed scheme. Unlike previous work, the new scheme satisfies all the desired properties.

TABLE I
SUMMARY OF CURRENT WATERMARKING SCHEMES

		Invisibility Level-I	Invisibility Level-II	Robustness	Blindness
Interval-based	[2]	no	no	yes	yes
	[8]	no	no	yes	yes
	[28]	yes	no	yes	yes
IPD-based	[6]	yes	yes*	no	yes
	[9]	yes	no	yes	no
	The proposed scheme	**yes**	**yes***	**yes**	**yes**

*The Level-II attack model is effective only when network jitter is small. Schemes like ours that add very small perturbations to IPDs remain hidden under normal network operating conditions (See §V-C).

III. NOTATIONS AND DEFINITIONS

In the discussion of the rest of the paper, we use the following notation. $\mathbf{a}^b = \{a_1, a_2, \cdots, a_b\}$ is a sequence of length b; $a_t^r = \{a_t, \cdots, a_r\}$ is a sequence in $\mathbf{a}^b$ starting with index r and ending with t. Specially, if $r \leq t$, a_t^r is an empty sequence, denoted by $\emptyset$; $\oplus$ denotes the 'xor' operation.

We also define the following variables used in our scheme.

- $\mathbf{I}^M$ is the IPD sequence of an original packet flow, where each delay, I_i, is positive real valued;
- $\mathbf{I'}^M$ is the IPD sequence of the same flow after injection of the watermark pattern;
- $\hat{\mathbf{I}}^{M'}$ is the IPD sequence received by the watermark decoder;
- $\mathbf{w}^n$ is the binary watermark sequence;
- $\tilde{\mathbf{w}}^N$ is a sparse version of $\mathbf{w}^N$, where $N = sn$;
- s is the *sparsification factor* and is integer valued;
- f is the density of $\tilde{\mathbf{w}}^N$ (see (2));
- $\mathbf{k}^N$ is a pseudo-random binary key sequence;
- $\mathbf{x}^N$ is a binary sequence, generated from the watermark $\mathbf{w}^n$ and the key $\mathbf{k}^N$, and embedded into flow IPDs;
- $\mathbf{y}^{N'}$ is decoder's estimate of $\mathbf{x}^N$;
- $\hat{\mathbf{w}}^n$ the estimate of the watermark sequence $\mathbf{w}^n$ at the decoder;
- Δ is a real-value step size used for IPD quantizations. It represents the strength of the watermark signal;
- σ is the standard deviation of jitter;

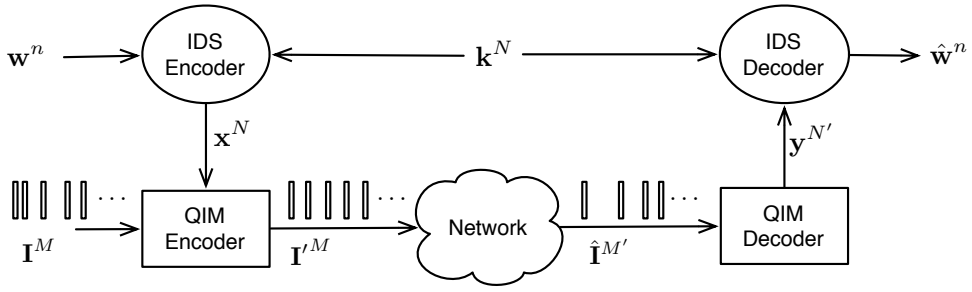


Fig. 3. An overview of the proposed flow watermarking scheme. The watermark sequence $\mathbf{w}^n$ is first transformed into a codeword $\mathbf{x}^N$ with the help of the key $\mathbf{k}^N$. $\mathbf{x}^N$ is then embedded into flow IPDs using QIM. At the decoder, the IPDs are processed by a QIM decoder to extract the codeword $\mathbf{y}^{N'}$, from which the IDS decoder recovers the watermark $\hat{\mathbf{w}}^N$, subsequently.

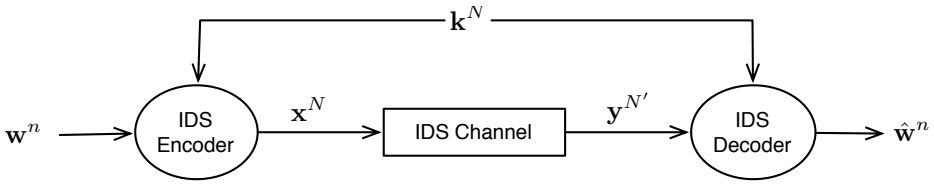


Fig. 4. Abstraction of communication channel. The IDS encoder/decoder pair help correct the dependent substitution, deletion, and bursty insertion errors on the channel.

- P_s, P_I, and P_d represent the probability of a substitution, an insertion, and a deletion event in the communication channel model of the network, respectively.

IV. The Proposed Scheme

A. Overview of the System

Figure 3 depicts the schematic of our proposed scheme, which can be divided into two layers: the insertion deletion substitution (IDS) encoder/decoder and the quantization index modulation (QIM) encoder/decoder. In the upper layer, the watermark sequence $\mathbf{w}^n$ is processed to generate an IDS error-correction codeword $\mathbf{x}^N$. On the lower layer, a QIM framework is used to inject $\mathbf{x}^N$ into the IPDs of the flow. QIM embedding is blind and causes little change to packet timings [12]. Upon receiving the flow, the QIM decoder extracts the pattern $\mathbf{y}^{N'}$. Subsequently an IDS decoder recovers the watermark, $\hat{\mathbf{w}}^n$, from this pattern.

If we abstract the QIM encoder, the network, and the QIM decoder together as a channel, which takes $\mathbf{x}^N$ as the input and spits out $\mathbf{y}^{N'}$, flow watermarking is equivalent to solving the problem of sending

one bit of information (the presence of the watermark) over this compound communication channel (see Figure 4). Codes for this compound channel must withstand *dependent substitution, deletion, and bursty insertion* errors. We next introduce each component of our scheme in details.

B. Insertion Deletion Substitution (IDS) Encoder

Our IDS error correction scheme is inspired by [25], [26], where a 'marker' code is employed to provide reliable communications over a channel with deletion and insertion errors. However, the approach in [25], [26] is not directly applicable to our channel, as we need to deal with somewhat more complicated errors, such as dependent substitution, deletion, and bursty insertions which we discuss in §IV-C2.

The IDS encoder works as follows. The watermark sequence $\mathbf{w}^n$ is first sparsified into a longer sequence $\tilde{\mathbf{w}}^N$ of length $N = sn$, as given by

$$\tilde{w}^{js}_{(j-1)s+1} = S\left(w_j\right), \quad j = 1, 2, \cdots, n, \tag{1}$$

where $S(\cdot)$ is a deterministic sparsification function that pads w_j with zeros, and is known at the decoder. We denote by density f the ratio of '1' in $\tilde{\mathbf{w}}^N$, i.e.,

$$f = \frac{\sum_{i=1}^{N} \tilde{w}_i}{N}. \tag{2}$$

f is a decoding parameter shared with the IDS decoder. The sparsified watermark $\tilde{\mathbf{w}}^N$ is then added to a key $\mathbf{k}^N$ to form the codeword $\mathbf{x}^N$:

$$x_i = \tilde{w}_i \oplus k_i, \quad i = 1, 2, \cdots, N, \tag{3}$$

where $\mathbf{k}^N$ is pseudo-random *key* sequence which is also known at the decoder.

Let us work on a small example of embedding one bit of watermark $w_1 = 1$ in a length 8 sequence. First w_1 is sparsified into an 8 bit sequence $\tilde{\mathbf{w}}^8 = 10000000$ (the sparsification factor $s = 8$). Then we add this sparse sequence to the first 8 bit of our key, $\mathbf{k}^8 = 11111011$. The resulting codeword is $\mathbf{x}^8 = 01111011$. Because $\mathbf{x}^8$ is only different from the key at one position, the decoder could infer the positions of deleted or inserted bits by comparing the received codeword with the key. For instance, if the decoder receives a codeword $\mathbf{y}^7 = 0111011$, one bit shorter than the key, then it knows that most likely a bit '1' from the second run was lost during transmission. Based on this observation, a probabilistic

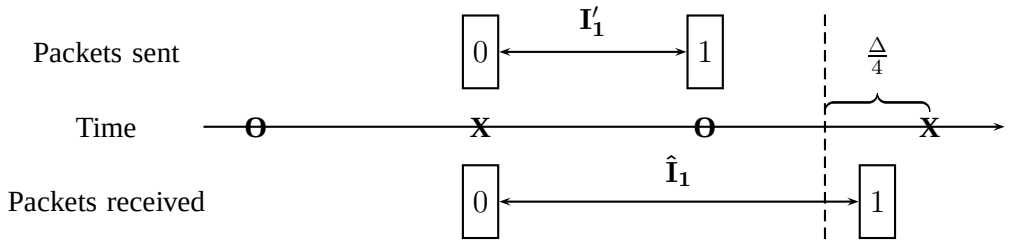


Fig. 5. An example of substitution errors caused by network jitter. '$\mathbf{x}$'s denote even quantizers and '$\mathbf{o}$'s odd quantizers. The bit embedded in I_1 is '1', but the decoded bit from $\hat{I}_1$ (delay between received packets 0 and 1) is '0'.

decoder can be developed to fully recover embedded bits, as will be discussed in §IV-D.

Since $\hat{\mathbf{w}}^N$ is sparse, the codeword $\mathbf{x}^N$ is close to the key, which is known at the IDS decoder. Therefore, the IDS encoding helps synchronize the lost/inserted bits at the cost of information capacity over the channel, which is not a concern for flow watermarking (see §II-B).

C. Insertion Deletion Substitution (IDS) Channel

1) QIM Embedding: The codeword $\mathbf{x}^N$ is injected into IPDs of the original flow using QIM embedding. Given a quantization step size Δ, the QIM encoder changes the IPD, I_i, into an even (or odd) multiplier of $\frac{\Delta}{2}$ given the embedded bit x_i is a bit 0 (or 1). The IPDs after modifications are given by

$$I'_i = \begin{cases} \left\lceil \frac{\max\left(\sum_{j=1}^{i} I_j - \sum_{j=1}^{i-1} I'_j, 0\right)}{\Delta} \right\rceil \Delta & \text{if } x_i = 0, \\ \left(\left\lceil \frac{\max\left(\sum_{j=1}^{i} I_j - \sum_{j=1}^{i-1} I'_j, 0\right)}{\Delta} \right\rceil + 0.5 \right) \Delta & \text{if } x_i = 1, \end{cases} \tag{4}$$

for $i = 1, 2, \cdots N$, where the ceiling function describes the operation that adds minimum delays to $Packet\ i$ to form the desired multiplier of $\frac{\Delta}{2}$.

At the QIM decoder, each embedded bit is extracted based on whether a received IPD is closer to an even or odd quantizer, i.e.,

$$y_i = \begin{cases} \lfloor \frac{2\hat{I}_i}{\Delta} \rfloor \mod 2 & \text{if } \frac{2\hat{I}_i}{\Delta} - \lfloor \frac{2\hat{I}_i}{\Delta} \rfloor \leq 0.5, \\ \lceil \frac{2\hat{I}_i}{\Delta} \rceil \mod 2 & \text{if } \frac{2\hat{I}_i}{\Delta} - \lfloor \frac{2\hat{I}_i}{\Delta} \rfloor > 0.5. \end{cases} \tag{5}$$

2) Channel Model: In presence of network artifacts, received IPDs, $\hat{\mathbf{I}}^{M'}$, are different from the original IPDs $\mathbf{I}^M$, leading to errors in decoding $\mathbf{x}^N$. *Substitution* errors occur when network jitter alters IPDs

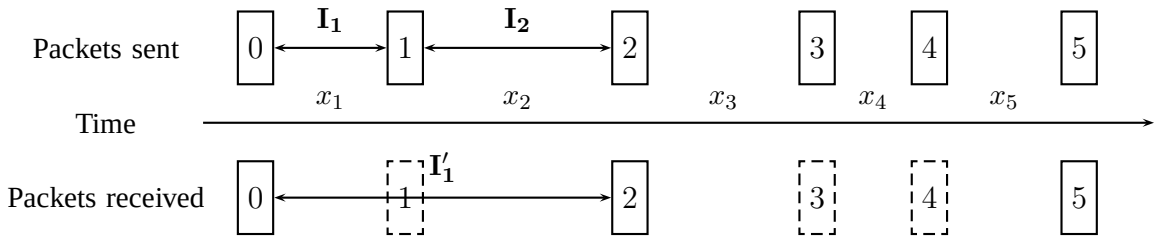


Fig. 6. Merging of IPDs as the result of packet drops. The deletion of Packet 1 merges the first two IPDs I_1 and I_2. and the deletions of Packet 3 and 4 merge I_3, I_4 and I_5.

largely. Figure 5 depicts one example where an embedded bit is flipped by jitter. In Figure 5, the bit '$x_1 = 1$' was originally encoded in the IPD I_1, resulting in $I'_1 = \frac{\Delta}{2}$. But at the QIM decoder, the received IPD $\hat{I}_1$ is pushed by jitter into interval $(\frac{3\Delta}{4}, \Delta)$, and thus decoded as '$y_1 = 0$'. In absence of packet drops or splits, a watermark bit flips if the IPD jitter is larger than $\frac{\Delta}{4}$. Following the observation of previous work that shows IPD jitter (within a certain period of time) is approximately i.i.d. zero-mean Laplace distributed [9], the probability of a substitution error by jitter can be estimated as

$$P_s = 1 - F\left(\frac{\Delta}{4}\right) = \frac{1}{2} e^{\frac{-\Delta}{2\sqrt{2}\sigma}}, \tag{6}$$

where $F(\cdot)$ is the Laplacian pdf and σ^2 is its variance.

Decoding errors also occur when packets are dropped. As packet drops lead to the merger of successive IPDs, the resulting error contains both deletion and substitution, which we refer to as *dependent deletion and substitution error*. For instance in Figure 6, deletion of Packet 1 merges the IPDs I_1 and I_2 into a large received IPD I'_1. As a result, instead of x_1 and x_2, only one bit $x_1 \oplus x_2$ is received at the decoder. We consider this case as a deletion of x_1, and possibly a substitution of x_2. In this paper, we assume that each packet is dropped independently with probability P_d. For the convenience of analysis, we also assume that the head of watermarked packet sequence, $Packet\ 0$, is not dropped.

The last type of error comes from packet insertions. This happens when packets are split to meet a smaller packet size limit, or when TCP transmission is triggered by network congestions. Both cases cause bursty insertions of packets. An example of such a scenario is depicted in Figure 7. Packet 2 is

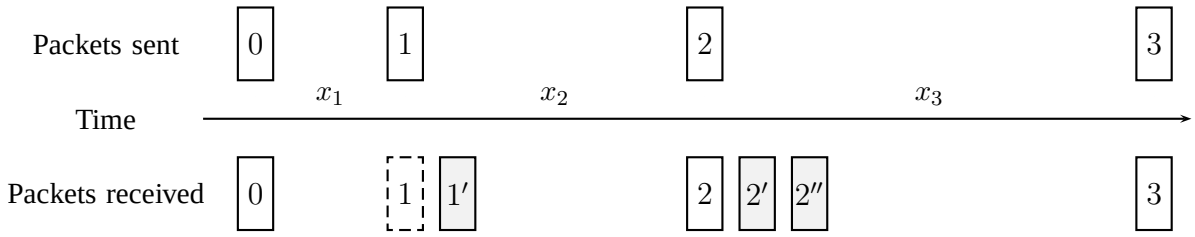


Fig. 7. A scenario with packet insertions. Packet 1 is split into two packets, and Packet 2 is split into three pieces.

split into three smaller ones, creating two new IPDs (2-2' and 2'-2", both with zero length). Therefore, two extra '0' bits would be decoded in $\mathbf{y}^{N'}$. In general, newly generated packets are mostly right next to the original one, hence we consider all inserted bits are'0's.[2] Furthermore, we assume the number of inserted packets follows a geometric distribution with parameter P_I.

D. Insertion Deletion Substitution (IDS) Decoder

We estimate each watermark bit from $\mathbf{y}^{N'}$ using the maximum likelihood decoding rule given by

$$\hat{w}_j = \arg \max_{w_j \in \{0,1\}} P\left(\mathbf{y}^{N'} | w_j\right), \quad j = 1, 2, \cdots, n. \tag{7}$$

Since $\mathbf{x}^N$ is a deterministic function of $\mathbf{w}$, we derive the likelihood in (7) based on the dependency between $\mathbf{y}^{N'}$ and $\mathbf{x}^N$ over the IDS channel. Suppose the QIM decoder received $i'-1$ packets after the first $i-1$ packets were sent out by the QIM encoder, and assume the $i'-1^{th}$ packet in the received flow corresponds to the q^{th} packet in the sent flow or a packet inserted immediately after it ($q \leq i-1$). The possible outcomes after $Packet\ i$ is sent are:

- if $Packet\ i$ in the sent flow is lost and no packets are inserted, the QIM decoder cannot decode new bits;
- if $Packet\ i$ is lost but $l > 0$ new packets are inserted right after it, the decoder could decode l bits, $y_{i'}^{i'+l-1}$, from newly received IPDs;

[2]Our methodology can be extended to cover the case that both '0' and '1' bits may be inserted.

- if $Packet\ i$ is received and additionally $l \geq 0$ packets are inserted, the decoder can decode $l+1$ new bits, $y_{i'}^{i'+l}$.

In the last two cases, the new IPD between the $i'-1^{th}$ packet and i'^{th} packets in the received flow corresponds to the merger of all IPDs between $Packet\ q$ and $Packet\ i$ in the sent flow. Hence, the first new bit, $y_{i'}$, is given by

$$y_{i'} = \begin{cases} x_{q+1} \oplus x_{q+2} \cdots \oplus x_i & \text{w. p.} \quad 1-P_s, \\ x_{q+1} \oplus x_{q+2} \cdots \oplus x_i \oplus 1 & \text{w. p.} \quad P_s, \end{cases} \tag{8}$$

where P_s is the probability of a substitution error given in (6). The remaining new bits $y_{i'+1}^{i'+l-1}$ (or $y_{i'+1}^{i'+l}$) are just '0' bits resulting from bursty packet insertions.

1) Hidden Markov Model: To capture the evolution of newly decoded bits from the received flow, we define the state after sending each packet with the pair (x_i', d_i), for $i = 1, 2, \cdots N$, where

- The *accumulated bit* x_i' it the sum of all bits resulting from merger of the IPDs between $Packet\ i$ and the previous packet that was received at the decoder. If $Packet\ i-1$ was received, then the x_i' is just the bit embedded on the IPD between $Packet\ i$ and $i-1$, i.e., x_i. On the other hand, if $Packet\ i-1$ was completely lost (i.e., after its deletion, there were no insertions), x_i' would be the sum of current bit x_i and bits embedded on previously merged IPDs, i.e., $x_i \oplus x_{i-1}'$. To sum up,

$$x_i' = \begin{cases} x_i & \text{w. p.} \quad 1 - P_d\left(1-P_I\right), \\ x_i \oplus x_{i-1}' & \text{w. p.} \quad P_d\left(1-P_I\right). \end{cases} \tag{9}$$

  Recall from (3) that $\mathbf{x}^N$ is generatd using the key $\mathbf{k}^N$ and the sparse watermark sequence $\tilde{\mathbf{w}}$. We will model the sparse watermark bits $\tilde{w}_i$'s as independent $Bernoulli(f)$ random variables. Therefore (9) can be rewritten as

$$x_i' = \begin{cases} k_i & \text{w. p.} \quad (1-f)\left(1-P_d(1-P_I)\right), \\ k_i \oplus 1 & \text{w. p.} \quad f\left(1-P_d(1-P_I)\right), \\ k_i \oplus x_{i-1}' & \text{w. p.} \quad (1-f)P_d\left(1-P_I\right), \\ k_i \oplus x_{i-1}' \oplus 1 & \text{w. p.} \quad fP_d\left(1-P_I\right). \end{cases} \tag{10}$$

Note that from (9), we can rewrite (8) as

$$y_{i'} = \begin{cases} x'_i & \text{w. p.} \quad 1 - P_s, \\ x'_i \oplus 1 & \text{w. p.} \quad P_s. \end{cases} \tag{11}$$

- The *drift* d_i is the shift in position of the sent $Packet\ i$ in the received flow, i.e., if $Packet\ i$ was not lost, it would appear at position $i' = i + d_i$ in the received flow. Given d_{i-1}, the drift of $Packet\ i$ is updated as

$$d_i = \begin{cases} d_{i-1} - 1 & \text{w. p.} \quad P_d\left(1 - P_I\right), \\ d_{i-1} + l, l \geq 0 & \text{w. p.} \quad \left(P_d P_I^{l+1}(1 - P_I) + (1 - P_d)P_I^l(1 - P_I)\right), \end{cases} \tag{12}$$

  where the first case occurs when $Packet\ i-1$ was dropped with no new packets inserted, and the second case occurs when total of l packets are received either because $Packet\ i-1$ was dropped and there were $l+1$ insertions or $Packet\ i$ was received and there were l insertions. For the first Packet 0, we initialize $d_0 = 0$. This minor change let us relax the synchronization requirement on the first packet.

Combine (11) and (10), and given $i' = i + d_i$, we have

$$y_{i+d_i} = \begin{cases} k_i & \text{w. p.} \quad \left((1-f)(1-P_s) + fP_s\right)\left(1 - P_d(1 - P_I)\right), \\ k_i \oplus 1 & \text{w. p.} \quad \left(f(1-P_s) + (1-f)P_s\right)\left(1 - P_d(1 - P_I)\right), \\ x'_{i-1} \oplus k_i & \text{w. p.} \quad \left((1-f)(1-P_s) + fP_s\right)P_d\left(1 - P_I\right), \\ x'_{i-1} \oplus k_i \oplus 1 & \text{w. p.} \quad \left(f(1-P_s) + (1-f)P_s\right)P_d\left(1 - P_I\right). \end{cases} \tag{13}$$

Equation (13) captures the HMM with hidden states of $(x'_i, d_i), i = 1, 2, \cdots N$ and observation states of $\mathbf{y}^{N'}$, as depicted in Figure 8. The state transition probabilities $P\left(y_{i-1+d_{i-1}}^{i-1+d_i}, x'_i, d_i | x'_{i-1}, d_{i-1}\right)$ can be

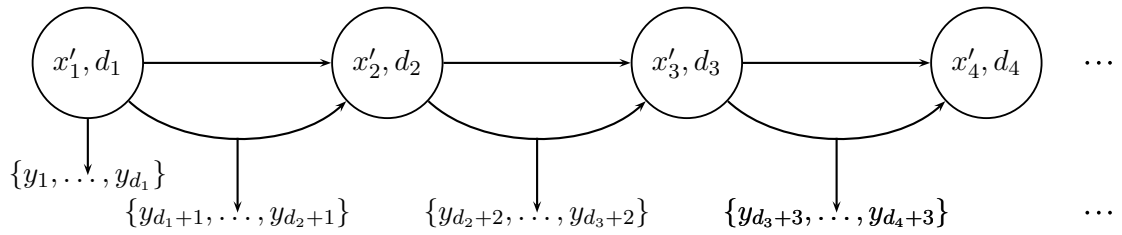


Fig. 8. The hidden-Markov model for the IDS channel. The observations are the codewords (y_i's) received by the IDS decoder, and the hidden states keep track of the drift and accumulated bit when sending every packet.

derived using (10), (12) and (13), summarized as

$$P\left(y_{i-1+d_{i-1}}^{i-1+d_i}, x'_i, d_i | x'_{i-1}, d_{i-1}\right) =$$

$$\begin{cases} (1-f)P_d(1-P_I) & \text{if } x'_i = x'_{i-1} \oplus k_i, d_i = d_{i-1} - 1 \text{ and } y_{i-1+d_{i-1}}^{i-1+d_i} = \emptyset, \\ fP_d(1-P_I) & \text{if } x'_i = x'_{i-1} \oplus k_i \oplus 1, d_i = d_{i-1} - 1 \text{ and } y_{i-1+d_{i-1}}^{i-1+d_i} = \emptyset, \\ (1-f)(1-P_s)(1-P_I)(P_d P_I^{l+1} + (1-P_d)P_I^l) & \text{if } x'_i = k_i, d_i = d_{i-1} + l \text{ and } y_{i-1+d_{i-1}} = x'_{i-1}, \\ f(1-P_s)(1-P_I)(P_d P_I^{l+1} + (1-P_d)P_I^l) & \text{if } x'_i = k_i \oplus 1, d_i = d_{i-1} + l \text{ and } y_{i-1+d_{i-1}} = x'_{i-1}, \\ (1-f)P_s(1-P_I)(P_d P_I^{l+1} + (1-P_d)P_I^l) & \text{if } x'_i = k_i, d_i = d_{i-1} + l \text{ and } y_{i-1+d_{i-1}} = x'_{i-1} \oplus 1, \\ fP_s(1-P_I)(P_d P_I^{l+1} + (1-P_d)P_I^l) & \text{if } x'_i = k_i \oplus 1, d_i = d_{i-1} + l \text{ and } y_{i-1+d_{i-1}} = x'_{i-1} \oplus 1. \end{cases} \tag{14}$$

For example, after sending $Packet\ i-1$, the system state is (x'_{i-1}, d_{i-1}). If $Packet\ i-1$ is lost and no packets are inserted. Then from (12), the drift of $Packet\ i$ becomes $d_i = d_{i-1} - 1$, and no new bit is decoded, i.e., $y_{i-1+d_{i-1}}^{i-1+d_i}$ is an empty sequence. Additionally, the IPD between $Packet\ i$ and $i-1$ is added to previously merged IPDs such that x'_i is decided based on the last two cases in (10). Overall, the transition probability in this scenario is given by

$$P\left(\emptyset, x'_i, d_{i-1} - 1 | x'_{i-1}, d_{i-1}\right) = \begin{cases} (1-f)P_d(1-P_I) & \text{if} \quad x'_i = x'_{i-1} \oplus k_i, \\ fP_d(1-P_I) & \text{if} \quad x'_i = x'_{i-1} \oplus k_i \oplus 1. \end{cases} \tag{15}$$

2) Forward-Backward Algorithm: For the HMM in Figure 8, we apply the forward-backward algorithm to derive the posterior probabilities $P(\mathbf{y}^{N'}|w_j)$, $j = 1, 2, \cdots, n$. Let us define the *forward* quantity as the

joint probability of bits $y_1^{i-1+d_i}$ decoded before sending $Packet\ i$ at the hidden state of (x_i', d_i), which is given by

$$F_i(x_i', d_i) = P(y_1^{i-1+d_i}, x_i', d_i), \quad i = 1, 2, \cdots, N. \tag{16}$$

The forward quantities can be computed recursively using transition probabilities in (14) as

$$F_i(x_i', d_i) = \sum_{\substack{x_{i-1}', \\ d_{i-1}}} F_{i-1}(x_{i-1}', d_{i-1}) P(y_{i-1+d_{i-1}}^{i-1+d_i}, x_i', d_i | x_{i-1}', d_{i-1}). \tag{17}$$

Similarly, we define the *backward* quantity as the conditional probability of decoding the rest of the bits in the received flow, $y_{i+d_i}^{N'}$, given the current state (x_i', d_i),

$$B_i(x_i', d_i) = P(y_{i+d_i}^{N'} | x_i', d_i), \quad i = 1, 2, \cdots, N. \tag{18}$$

The backward quantities can also be computed recursively as

$$B_i(x_i', d_i) = \sum_{\substack{x_{i+1}', \\ d_{i+1}}} P(y_{i+d_i}^{i+d_{i+1}}, x_{i+1}', d_{i+1} | x_i', d_i) B_{i+1}(x_{i+1}', d_{i+1}). \tag{19}$$

Given the forward/backward quantities, the posterior likelihood of the watermark bit w_j is given by

$$\begin{aligned} P\left(\mathbf{y}^{N'} | w_j\right) &= P\left(\mathbf{y}^{N'} | \tilde{w}_{(j-1)s+1}^{js}\right) \\ &= \sum_{\substack{x_{(j-1)s}', x_{js}', \\ d_{(j-1)s}, d_{js}}} F_{(j-1)s}\left(x_{(j-1)s}', d_{(j-1)s}\right) \hat{F}_{js}\left(x_{js}', d_{js}\right) B_{js}(x_{js}', d_{js}), \end{aligned} \tag{20}$$

where the first equality follows from our watermark sparsification function in (1), and the quantity $F_{js}'(x_i, d_i)$ is defined as

$$\hat{F}_{js}(x_i', d_i) = P\left(y_{(j-1)s+d_{(j-1)s}}^{i-1+d_i}, x_i', d_i | x_{(j-1)s}', d_{(j-1)s}, \tilde{w}_{(j-1)s+1}^{js}\right), \quad (j-1)s + 1 \le i \le js. \tag{21}$$

The quantity $F_{js}'(x_i', d_i)$ can be calculated recursively as

$$\hat{F}_{js}(x_i', d_i) = \sum_{\substack{x_{i-1}', \\ d_{i-1}}} \hat{F}_{js}(x_{i-1}', d_{i-1}) P\left(y_{i-1+d_{i-1}}^{i-1+d_i}, x_i', d_i | x_{i-1}', d_{i-1}, \tilde{w}_{(j-1)s+1)}^{js}\right), \tag{22}$$

TABLE II
TRUE POSITIVE RATES WITH VARYING WATERMARK PARAMETERS WHEN FALSE POSITIVE RATE IS FIXED BELOW 1%.

n \ Δ (ms)	20	60	100
10	0.0310	0.6050	0.6224
30	0.0310	0.9790	0.9970
50	0.0272	0.9990	1

TABLE III
TRUE POSITIVE RATES UNDER VARYING IPD JITTER WITH FALSE POSITIVE RATES BELOW 1%.

Jitter Std. Dev. (ms)	10	20	30	40
Synthetic traffic	1.000	0.989	0.770	0.232
Real traffic	1.000	0.989	0.652	0.193

where $P\left(y_{i-1+d_{i-1}}^{i-1+d_i}, x_i', d_i | x_{i-1}', d_{i-1}, \tilde{w}_{(j-1)s+1)}^{js}\right)$ is given by

$$P\left(y_{i-1+d_{i-1}}^{i-1+d_i}, x_i', d_i | x_{i-1}', d_{i-1}, \tilde{w}_{(j-1)s+1)}^{js}\right) =$$

$$\begin{cases} P_d(1-P_I) & \text{if } d_i = d_{i-1}-1 \text{ and } x_i' = \tilde{w}_i \oplus k_i \oplus x_{i-1}', \text{ and } y_{i-1+d_{i-1}}^{i-1+d_i} = \emptyset, \\ P_s(1-P_I)\left(P_d P_I^{d_i-d_{i-1}+1} + (1-P_d)P_I^{d_i-d_{i-1}}\right) & \text{if } d_i \geq d_{i-1}, x_i' = \tilde{w}_i \oplus k_i \text{ and } y_{i-1+d_{i-1}} = x_{i-1}' \oplus 1, \\ (1-P_s)(1-P_I)\left(P_d P_I^{d_i-d_{i-1}+1} + (1-P_d)P_I^{d_i-d_{i-1}}\right) & \text{if } d_i \geq d_{i-1}, x_i' = \tilde{w}_i \oplus k_i \text{ and } y_{i-1+d_{i-1}} = x_{i-1}'. \end{cases} \tag{23}$$

Once the posterior probabilities for all watermark bits are calculated, the watermark sequence, $\hat{\mathbf{w}}^n$ can be estimated using maximum likelihood rule of (7). Finally, the presence of the watermark in a flow is decided based on the correlation value of the estimated watermark, $\hat{\mathbf{w}}^n$, and the original watermark sequence, $\mathbf{w}^n$.

V. EVALUATION

We tested our scheme for two groups of traces: *synthetic* packet flows of length 2000 generated from Poisson process with average rate of 3.3 packets per second (pps), and real *SSH* traces of length 2000 collected in CAIDA database with average rate of 0.865 pps [27], which represent typical traffic in human-involved network connections, where flow watermarks are most applicable.

TABLE IV
TRUE POSITIVE RATES FOR VARYING P_d WITH FALSE POSITIVE RATES BELOW 1%.

P_d	1%	2%	3%	10%
Synthetic traffic	1.000	1.000	1.000	0.995
Real traffic	1.000	1.000	1.000	0.996

TABLE V
TRUE POSITIVE RATES FOR VARING P_I WITH FALSE POSITIVE RATES BELOW 1%.

P_I	1%	5%	10%	20%
Synthetic traffic	1.000	1.000	1.000	0.500
Real traffic	1.000	1.000	1.000	0.568

TABLE VI
TRUE POSITIVE RATES FOR VARYING P_I,P_d WITH FALSE POSITIVE RATES BELOW 1%.

P_i,P_d	1%,1%	5%,5%	10%,10%
Synthetic	1.000	1.000	0.764
Real	1.000	1.000	0.662

A. Parameter Selection

The first test examined the effects of watermark length n and IPD quantization step size Δ. We varied n over $\{10, 30, 50\}$, Δ over $\{20, 60, 100\}$ ms and fixed the sparisificatoin factor $s = 10$. The deletion and insertion probabilities and the network jitter were set to $P_d = 0.1$, $P_I = 0$, $\sigma = 10$ ms, respectively. 5000 synthetic flows were embedded with watermarks and another 5000 unwatermarked ones served as the control group.

Table II shows the true positive rates of our test, when false positive rates were kept under 1%. As we increase watermark length or quantization step size (embed a 'stronger' pattern), detection error decreases.

For the tests in this section, we fix the watermark parameters to $\{\Delta = 100\,ms, n = 50, s = 10\}$, which had the best performance in Table II.

B. Robustness Tests

We evaluated watermark robustness against network jitter, and packet loss and insertion.

1) IPD jitter: From the experimental results in [28], the standard deviation of the Laplacian jitter is estimated as $\sigma = 10$ ms. We performed tests with σ varied over $\{10, 20, 30, 40\}$ ms. The packet drop and

TABLE VII
AVERAGE KS DISTANCES BETWEEN WATERMARKED AND UNWATERMARKED SYNTHETIC TRACES.

n \ Δ (ms)	100	80	60
30	0.0177	0.0138	0.0101
40	0.0233	0.0181	0.0133
50	0.0284	0.0223	0.0160

TABLE VIII
AVERAGE KS DISTANCES BETWEEN WATERMARKED AND UNWATERMARKED SSH TRACES.

n \ Δ (ms)	100	80	60
30	0.0091	0.0081	0.0071
40	0.0120	0.0111	0.0091
50	0.0158	0.0139	0.0123

split probabilities were $P_d = 0.1$ and $P_I = 0$, respectively. This time, we watermarked 1000 flows from both synthetic and SSH traces.

The true positive rates are given in Table III. Notice that the watermarks were detected with accuracies over 98%, even when jitter was as high as 20 ms. The detection performance falls sharply when jitter standard deviations exceeds 40 ms. However, such excessively large jitter rarely occurs at proper network conditions. Hence, our scheme withstands network jitter in normal operating conditions.

2) Packet deletion and insertion: One major improvement of our design over previous work is robustness against packet deletion and insertion. To verify this, we tested our scheme in a network with: solely packet deletion with probabilities $P_d = \{0.01, 0.02, 0.03, 0.1\}$, solely packet insertion with probabilities $P_I = \{0.01, 0.05, 0.1, 0.2\}$, and both deletion and insertion with probabilities $(P_d, P_I) = \{(0.01, 0.01), (0.05, 0.05), (0.1, 0.1), (0.15, 0.15)\}$. During all the tests, the standard deviation of jitter was fixed as $\sigma = 10\,ms$, and 1000 flows from both synthetic and SSH traces were used.

The results in Tables IV–VI demonstrate watermarks were detected with high accuracies when 5% of packets were dropped and inserted.

C. Visibility Tests

We first evaluated watermark invisibility with two Level-I attack tests: the Kolmogorov-Simirnov (KS) test and the multiflow attack (MFA) test.

TABLE IX
STATISTICS OF BLANK INTERVALS IN THE AGGREGATED FLOW FROM SYNTHETIC TRACES.

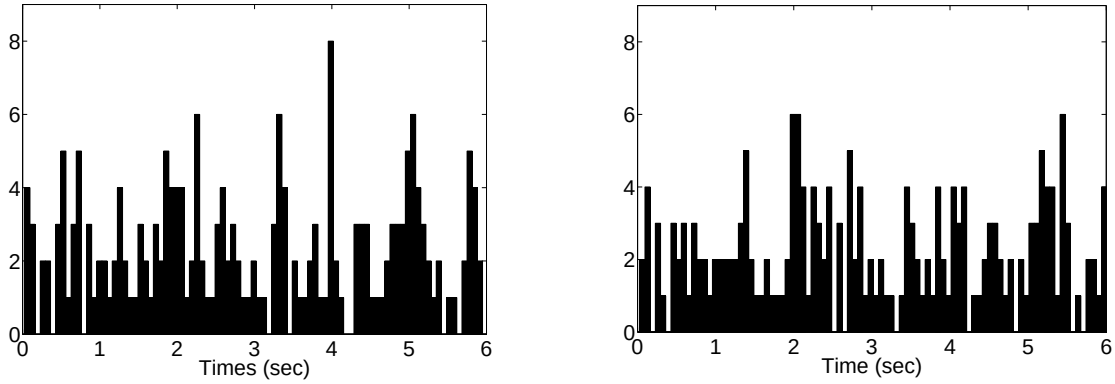
	Watermarked	Unwatermarked
Mean	24.07	25.96
Standard Deviation	5.246	5.187

TABLE X
STATISTICS OF BLANK INTERVALS IN THE AGGREGATED FLOW FROM SSH TRACES.

	Watermarked	Unwatermarked
Mean	403	395.67
Standard Deviation	20.54	16.84

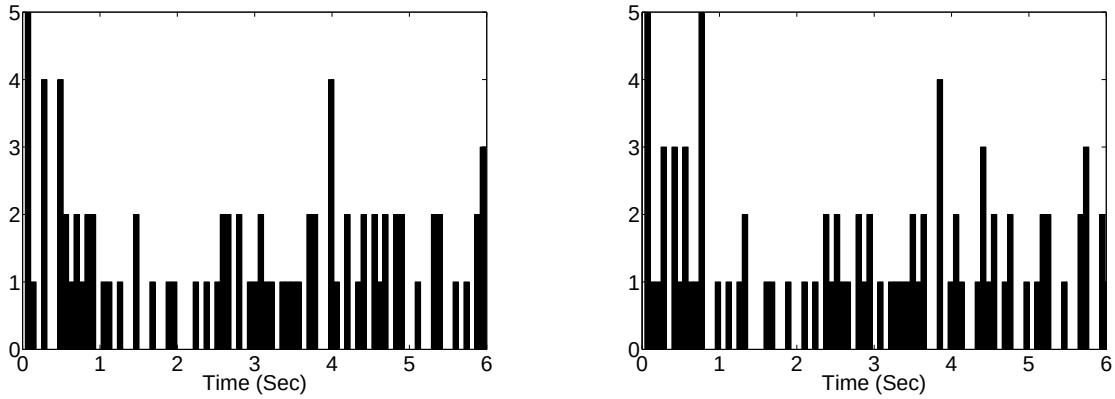
KS test is commonly applied to comparing distributions of datasets. Given two data sets, the KS distance is computed as the maximum difference of their empirical distribution functions [13]. For two flows A and B, the KS distance is given by $\sup_x(|F_A(x) - F_B(x)|)$, where $F_A(x)$ and $F_B(x)$ are the empirical pdfs of IPDs in A and B. We claim two flows are indistinguishable if their KS distance is below 0.036, a threshold suggested in [13]. We calculated the average KS distance between watermarked and unwatermarked flows using both synthetic and SSH traces. The results are tabulated in Tables VII and VIII. None of the KS distances exceed the detection threshold of visibility, which implies the embedded watermarks did not cause noticeable artifacts in the original packet flows.

MFA is a watermark attack that detects positions of embedded watermarks in interval-based schemes, When flows which were watermarked using the same watermark are aggregated, the aggregate flow shows a number of intervals containing no packets (see Figure 10 in [11]). To test whether such 'visible' pattern exists in flows watermarked using our scheme, we combined 10 watermarked and 10 unwatermarked flows for both the synthetic and SSH traces, and divided the aggregated flows into intervals with length of length of 70 ms. We then counted the number of blank intervals with no packets in each aggregate flow. This procedure was repeated 1000 times, and the resulting blank interval statistics are shown in Tables IX and X. For both synthetic and SSH traces, we see that the number of blank intervals does not change much after watermarks were embedded. Figure 9 depicts packet counts in each interval of length 70 ms in the aggregated synthetic traces. Comparing Figures 9(a) with 9(b), no clear watermark pattern is observed. The same observation was made in Figure 10, which depicts packet counts of SSH traces. Therefore, our scheme is resistant to MFA.



(a) Unwatermarked flows (b) Watermarked flows

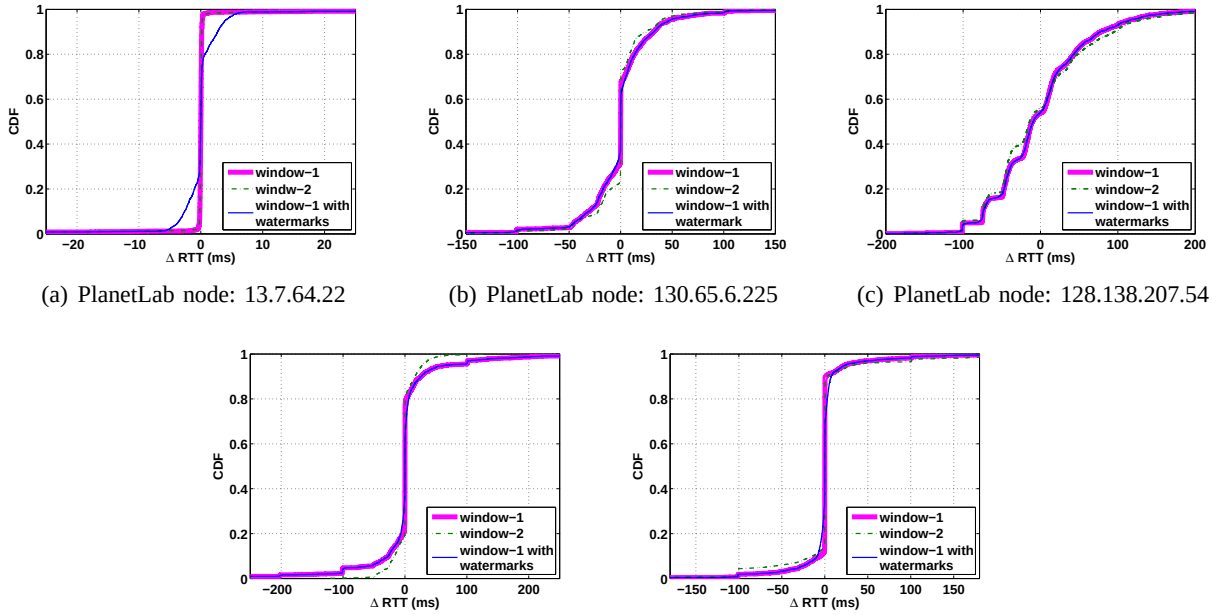
Fig. 9. MFA tests on synthetic traces: packet counts in intervals of 70 ms in the flow aggregated from (a) 10 unwatermarked flows and (b) 10 watermarked flows.



(a) Unwatermarked flows (b) Watermarked flows

Fig. 10. MFA tests on real SSH traces: packet counts in intervals of 70 ms in the flow aggregated from (a) 10 unwatermarked flows and (b) 10 watermarked flows.

We next tested the performance of our watermarks under a Level-II attack, BACKLIT, where the attacker sees both directions of a TCP connection [24]. BACKLIT detects watermarks in SSH flows based on the differences in round trip times (RTTs) of consecutive TCP requests, ΔRTT. We considered a stepping stone detection scenario in our campus network. Network jitter in such an environment, like most enterprise networks, is very small. According to our measurements from our lab machine to the campus exit node, the jitter standard deviation was as low as $\delta = 1.6$ ms. For this level of noise, a small quantization step size of IPDs, 10 ms, was sufficient to achieve accurate decoding performances



(a) PlanetLab node: 13.7.64.22 (b) PlanetLab node: 130.65.6.225 (c) PlanetLab node: 128.138.207.54

(d) PlanetLab node: 192.12.33.100 (e) PlanetLab node: 206.117.37.6

Fig. 11. Comparisons of the ΔRTT distributions before and after adding watermarking.

(true positive rate of 100% and false positive rate of less than 1%). We then examined the effect of our watermarks on the ΔRTT distribution of a SSH connection, by monitoring the RTT jitter from our lab to 5 PlanetLab nodes [29]. For each node, we issued 4000 ping requests with ping interval of 100 ms, and divided ping packets into two windows, each consisting of 2000 packets. We transplanted delays of SSH packets during watermarking onto the ping replies in Window-1, to mimic the effects of watermarking live TCP requests, and left Window-2 untouched as the control group. Figure 11 depicts the CDFs of ΔRTT of Window-1, Window-1 with watermarks and Window-2. We notice that except in Figure 11(a), where the RTT jitter to the destination node was extremely low, the watermarked flow is not distinguishable from the unwatermarked flow. Our results indicate that BACKLIT only works in "clean" environments with negligible jitter, and the subtle watermark we inject remains invisible when moderate jitter exists.

To achieve simultaneous watermark robustness and invisibility, we embed a sparse watermark using the QIM embedding into flow IPDs. Modeling the network jitter, deletions, and insertions as a communication channel descried by a HMM, and employing an IDS decoder, we can reliably decoder the watermark. The QIM embedding meanwhile guarantees that watermark remains invisible to attackers.

REFERENCES

[1] X. Gong, M. Rodrigues, and N. Kiyavash, "Invisible flow watermarks for channels with dependent substitution and deletion errors," in *IEEE International Conf. Acoustics, Speech and Signal Processing (ICASSP)*, Kyoto, Japan, 2012, pp. 1773–1776.

[2] X. Wang, S. Chen, and S. Jajodia, "Network flow watermarking attack on low-latency anonymous communication systems," in *IEEE Symposium on Security and Privacy*, Oakland, CA, USA, 2007, pp. 116–130.

[3] S. Staniford-Chen and L. T. Heberlein, "Holding intruders accountable on the Internet," in *IEEE Symposium on Security and Privacy*, Oakland, CA, USA, 1995, pp. 39–49.

[4] Y. Zhang and V. Paxson, "Detecting stepping stones," in *USENIX Security Symposium*, Denver, CO, USA, 2000, pp. 171–184.

[5] K. Yoda and H. Etoh, "Finding a connection chain for tracing intruders," in *Proc. 6th European Symposium on Research in Computer Security*, London, UK, 2000, pp. 191–205.

[6] X. Wang and D. S. Reeves, "Robust correlation of encrypted attack traffic through stepping stones by manipulation of interpacket delays," in *Proc. 10th ACM Conf. on Computer and Communications Security (CCS)*, Washington, DC, USA, 2003, pp. 20–29.

[7] X. Wang, S. Chen, and S. Jajodia, "Tracking anonymous peer-to-peer voip calls on the Internet," in *Proc. 12th ACM conference Computer and Communications Security (CCS)*, Alexandria, VA, USA, 2005.

[8] Y. J. Pyun, Y. H. Park, X. Wang, D. S. Reeves, and P. Ning, "Tracing traffic through intermediate hosts that repacketize flows," in *26th IEEE International Conf. on Computer Communications (Infocom)*, Anchorage, AK, USA, 2007, pp. 634–642.

[9] A. Houmansadr, N. Kiyavash, and N. Borisov, "Rainbow: A robust and invisible non-blind watermark for network flows," in *Proc. of 16th Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, USA, 2009.

[10] W. Yu, X. Fu, S. Graham, D. Xuan, and W. Zhao, "Dsss-based flow marking technique for invisible traceback," in *IEEE Symposium on Security and Privacy*, Oakland, CA, USA, 2007, pp. 18–32.

[11] N. Kiyavash, A. Houmansadr, and N. Borisov, "Multi-flow attacks against network flow watermarking schemes," in *Proc. 17th Conf. on USENIX Security Symposium*, San Jose, CA, USA, 2008, pp. 307–320.

[12] B. Chen and G. W. Wornell, "Quantization index modulation: A class of provably good methods for digital watermarking and information embedding," *IEEE Trans. Inf. Theory*, vol. 47, no. 4, pp. 1423–1443, May 2001.

[13] E. J. Massey, "The kolmogorov-smirnov test for goodness of fit," *Journal of the American Statistical Association*, vol. 46, no. 253, pp. 68–78, 1951.

[14] RFC4252. The secure shell (SSH) authentication protocol. [Online]. Available: http://tools.ietf.org/html/rfc4252

[15] K. H. Yung, "Detecting long connection chains of interactive terminal sessions," in *Proc. 5th International Conf. Recent Advances in Intrusion Detection(RAID)*, Zurich, Switzerland, 2002, pp. 1–16.

[16] W. Ding and S.-H. Huang, "Detecting intruders using a long connection chain to connect to a host," in *IEEE International Conf. Advanced Information Networking and Applications (AINA)*, Biopolis, Singapore, 2011, pp. 121–128.

[17] R. P. Lippmann, K. W. Ingols, C. Scott, K. Piwowarski, K. J. Kratkiewicz, M. Artz, and R. K. Cunningham, "Evaluating and strengthening enterprise network security using attack graphs," MIT Lincoln Lab., Tech. Rep. PR-IA-2, Aug. 12, 2005.

[18] I2P anonymous network. [Online]. Available: http://www.i2p2.de/

[19] The Freenet project. [Online]. Available: https://freenetproject.org/

[20] GNUnet. [Online]. Available: https://gnunet.org/

[21] I. Cox, M. Miller, and J. Bloom, *Digital Watermarking*. Morgan Kaufmann Publisher, 2001.

[22] S. D. Servettot, C. I. Podilchuks, and K. Ramchandrant, "Capacity issues in digital image watermarking," in *International Conf. Image Processing (ICIP)*, Chicago, IL, USA, 1998, pp. 445–449.

[23] Z. Lin and N. Hoppers, "New attacks on timing-based network flow watermarks," in *Proc. of the 21st USENIX Conf. on Security symposium*, Bellevue, WA, USA, 2012, pp. 20–35.

[24] X. Luo, P. Zhou, J. Zhang, R. Perdisci, W. Lee, and R. Chang, "Exposing invisible timing-based traffic watermarks with backlit," in *Proc. of the 27th Annual Computer Security Applications Conference*, Orlando, FL, USA, 2011, pp. 197–206.

[25] M. C. Davey and D. J. C. MacKay, "Watermark codes: reliable communication over insertion/deletion channels," in *Proc. IEEE International Symposium in Information Theory (ISIT)*, Sorrento, Italy, 2000, pp. 477–477.

[26] M. C. Davey and D. J. C. Mackay, "Reliable communication over channels with insertions, deletions, and substitutions," *IEEE Trans. Information Theory*, vol. 47, no. 2, pp. 687–698, Feb. 2001.

[27] The Cooperative Association for Internet Data Analysis. [Online]. Available: http://www.caida.org/

[28] A. Houmansadr and N. Borisov, "Swirl: A scalable watermark to detect correlated network flows," in *Proc. of 16th Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, USA, 2011.

[29] PlanetLab: An open platform for developing, deploying, and accessing planetary-scale services. [Online]. Available: http://www.planet-lab.org/