Network Flow Watermarking Attack on Low-Latency Anonymous Communication Systems

Xinyuan Wang
Dept. of ISE
George Mason University
Fairfax, VA 22030, USA
xwangc@gmu.edu

Shiping Chen*
Sybase, Inc.
One Sybase Dr.
Dublin, CA 94568, USA
shiping.chen@sybase.com

Sushil Jajodia
CSIS
George Mason University
Fairfax, VA 22030, USA
jajodia@gmu.edu

Abstract

Many proposed low-latency anonymous communication systems have used various flow transformations such as traffic padding, adding cover traffic (or bogus packets), packet dropping, flow mixing, flow splitting, and flow merging to achieve anonymity. It has long been believed that these flow transformations would effectively disguise network flows, thus achieve good anonymity. In this paper, we investigate the fundamental limitations of flow transformations in achieving anonymity, and we show that flow transformations do not necessarily provide the level of anonymity people have expected or believed. By injecting unique watermark into the inter-packet timing domain of a packet flow, we are able to make any sufficiently long flow uniquely identifiable even if 1) it is disguised by substantial amount of cover traffic, 2) it is mixed or merged with a number of other flows, 3) it is split into a number subflows, 4) there is a substantial portion of packets dropped, and 5) it is perturbed in timing due to either natural network delay jitter or deliberate timing perturbation. In addition to demonstrating the theoretical limitations of low-latency anonymous communications systems, we develop the first practical attack on the leading commercial low-latency anonymous communication system. Our real-time experiments show that our flow watermarking attack only needs about 10 minutes active Web browsing traffic to "penetrate" the Total Net Shield service provided by www.anonymizer.com.

Our analytical and empirical results demonstrate that achieving anonymity in low-latency communication systems is much harder than we have realized, and current flow transformation based low-latency anonymous communication systems need to be revisited.

*The work of Shiping Chen was done when he was at George Mason University.

1 Introduction

Privacy and anonymity have become major concerns as we are increasingly dependent on the Internet in our daily lives. For example, people sometimes do not want others to know what Web sites they have visited. Under certain circumstances, people want to remain anonymous to the Web sites they have visited so that their personal interests can not be profiled by the Web sites. To address these privacy concerns, a number of anonymous communication systems (e.g. DC-Net [7] Anonymizer.com [1], Crowds [25], Onion Routing [24], Tor [9], Hordes [27], Web Mixes [3]) have been designed to provide anonymity to the communicating parties.

According to Pfitzmann and Waidner [23], there are three types of anonymities that can be provided by anonymous communication systems: *sender anonymity*, *receiver anonymity*, and *unlinkability of sender and receiver*. Sender anonymity means that the identity of the information sender is hidden, and receiver anonymity means that the identity of the information receiver is hidden. Unlinkability of sender and receiver refers to the property that the sender and receiver of a communication cannot be identified even if the sender and receiver are known to be of communicating with someone. Since anonymity is the state of lacking identity, anonymous communication can only be achieved by removing all the identifying characteristics from the anonymized network flows.

It's well known that encryption alone is not adequate to achieve anonymity. For example, various traffic analysis techniques [33, 32, 10, 31, 30] have been shown to be able to uniquely identify encrypted flows. These traffic analysis techniques can be used to link the encrypted flow to its original information sender and receiver, which would break the sender and receiver anonymity as well as the unlinkability of sender and receiver.

Traditional methods of achieving anonymity in commu-

2007 IEEE Symposium on Security and Privacy(SP'07)
0-7695-2848-1/07 $20.00 © 2007 IEEE

nication include using proxies [24, 25, 9], MIXes [6, 17, 3], and various other flow transformations such as adding cover traffic, packet dropping, flow mixing, flow splitting, and flow merging. Since these flow transformations drastically change the original network flow, it is generally believed that these flow transformations would remove most, if not all, identifying characteristics of the original flow and make it indistinguishable from some other independent network flow. For example, cover traffic has long been believed to be able to prevent the adversary from using traffic analysis to uniquely identify the covered flow and link the information sender and receiver. A number of works [16, 15, 12] have used cover traffic to achieve anonymity. In addition, Blum *et al.* [5] claimed that a packet flow would become indistinguishable from other independent packet flows if the ratio of the cover traffic added to the original flow reaches certain threshold. They further claimed that their hardness result regarding the traffic analysis holds true even if the adversary is active.

In this paper, we investigate the fundamental limitations of flow transformations in anonymizing packet flows by taking the role of active adversary. If we can uniquely identify a packet flow in spite of various flow transformations, we could link the anonymized packet flow to its original flow thus break the anonymity. We exploit one fundamental limitation of low-latency anonymous communication systems – low-latency anonymizing systems do not eliminate the packet timing correlation between the anonymized flow and the original flow. Therefore, there exists mutual information in the packet timing domain between the anonymized flow and the original flow. Such mutual information forms the very foundation for the unique identification and tracking the anonymized flow.

The key technique we use in our investigation is to transparently watermark the packet flow by slightly adjusting the timing of selected packets. If the embedded unique watermark survives various flow transformations, the watermarked network flow can be uniquely identified and thus linked to it original sender and receiver. This network flow watermarking technique can be used to attack low-latency anonymous communication systems without global monitoring capability. To break the unlinkability of sender and receiver, we only need to monitor and perturb the network flows to and from potential senders and receivers we would like to verify. For example, a malicious Web site could watermark the Web traffic returned to its visitors, and determine if some suspected user has visited its Web site by checking if that user has received the (potentially anonymized) watermarked traffic. With appropriate monitoring capability, our flow watermarking technique can also be used to attack the sender and receiver anonymity.

By developing a novel flow watermarking technique, we discover a rather surprising result on the inherent limits of flow transformations in anonymizing long network flows. Our analysis shows that adding cover traffic, dropping packets, mixing or merging with other flows, and splitting into multiple subflows, do not necessarily make a long network flow indistinguishable from other independent flows. In fact, a sufficiently long flow could be uniquely identified through our flow watermarking technique even if the amount of cover traffic added is many times more than the number of original packets. This result is in contrast to many people's intuition – ours included. Our claims are backed by extensive offline experimental results and real-time experimental results on a leading commercial anonymizing service. In particular, we were able to "penetrate" the Total Net Shield, the "ultimate solution in online identity protection" of www.anonymizer.com, with our flow watermarking technique. We only needed less than 11 minutes of active surfing traffic from www.usatoday.com to achieve virtually 100% true positive rate and less than 0.3% false positive rate at the same time in linking the sender and receiver of the Web traffic that was anonymized by the Total Net Shield of www.anonymizer.com. Our analytical and empirical results demonstrate that 1) the anonymity provided by low-latency anonymous communication systems is fundamentally limited, 2) there exists practical attack to break the anonymity provided by existing low-latency anonymous communication systems, and 3) existing low-latency anonymous communication systems need to be revisited.

The rest of this paper is organized as follows. In Section 2, we elaborate the relation between the network flow identification and anonymous communication and review common flow transformations used in low-latency anonymous communication systems. In Section 3, we present our interval centroid based flow watermarking scheme. In Section 4, we present and analyze a few key properties of the watermarking scheme. In Section 5, we empirically validate our findings through real-time experiments on www.anonymizer.com and offline simulations. In Section 6, we review related works. In Section 7, we conclude the paper.

2 Network Flow Identification and Anonymous Communication

In this section, we formulate the network flow identification problem in the context of network information flow, and elaborate on the relationship between the network flow identification and anonymous communication by reviewing the flow transformations used in existing anonymous communication systems.

2007 IEEE Symposium on Security and Privacy(SP'07)
0-7695-2848-1/07 $20.00 © 2007 IEEE

2.1 Network Information Flow and Network Flow Identification

A network generally has multiple network flows between different nodes. Some network flows are essentially correlated with each other in that they are part of the transmission of the same information. For example, multicast flows from the same source are essentially correlated if they convey the same information. All of the connections in a connection chain across stepping stones are essentially correlated since those connections have the same essential payload even if the payload is encrypted.

Here we use *network information flow* to represent the transmission path of some information along the network. Therefore, any communication between different nodes in a network, whether it has single or multiple sources/destinations, is a network information flow. A network information flow may consist of multiple network flows which may appear very different due to various flow transformations. As indicated by the multicast example, a network information flow is not necessarily linear.

A generic problem of network information flow is how to determine those network flows that belong to any particular network information flows. We define this problem as the *network flow identification problem*.

Network flow identification is inherently related to anonymous communication whose goal is to conceal the true identities and relationships among the communicating parties. For example, if we can identify and authenticate those network flows that belong to any particular network information flow, we will be able to link a network flow to its information source and destination. Thus, we can link the information sender and receiver.

2.2 Anonymous Communication and Transformations of Network Flow

To conceal the true identities and relationships among the communicating parties, anonymous communication systems usually mix multiple network information flows among multiple communicating parties and transform each network flow substantially. If the transformed network flows do not have any identifying characteristics that can be linked to their information sources or destinations, anonymity will be achieved.

Existing network flow transformations used by current anonymous communication systems can be broadly divided into two categories: *intra-flow transformations* and *inter-flow transformations*. The intra-flow transformations are those transformations that are within the boundary of the flow without involving any other flow during the transformation. Inter-flow transformations are those that involve more than one flow.

Figure 1 illustrates most common forms of intra-flow transformation: *adding chaff*, *packet dropping*, and *repacketization*. Here we assume that all the network flows have been encrypted, and we do not show the timing perturbation and packet reorder in the figure. Reordering of encrypted packets is equivalent to timing perturbation of encrypted packets from an outsider's point of view, and a number of works [33, 32, 10, 30, 29] have addressed the timing perturbation of encrypted flows. Here, chaff refers to any bogus packet added to the flow that was not part of the original flow. For example, any cover traffic used in anonymous communication systems [12, 15] is chaff. Packet dropping can happen naturally, but it can be introduced deliberately as an effort to achieve anonymity [19]. Repacketization can either combine two or more closely adjacent packets into a larger packet or split a packet into multiple smaller packets. Both forms of repacketization can occur naturally and be triggered deliberately. For example, SSH is known to combine closely adjacent packets into larger packets. IP fragmentation happens when the packet size is larger than the Maximum Transmission Unit (MTU) along the transmission path. Without considering the packet size, we can view two forms of repacketization as either chaff or de-chaff.

Figure 2 illustrates most common forms of inter-flow transformation: *flow mixing*, *flow splitting* and *flow merging*. Flow mixing refers to mixing some flow f_0 with some unrelated flows: $f_1, \ldots, f_n$ to generate mixed flow f_0'. To further frustrate any flow correlation, a flow f_0 could be split into multiple subflows: $f_0^1, \ldots, f_0^n$, which could be later merged. The difference between flow mixing and flow merging is that flow mixing combines a flow with unrelated flows, and flow merging combines a flow with flows that belong to the same network information flow. When all of the network flows are encrypted, flow mixing and flow merging appear the same. Furthermore, the flows $f_1, \ldots, f_n$ mixed with flow f_0 in Figure 2a (flow mixing) and flows $f_0^2, \ldots, f_0^n$ merged with flow f_0^1 in Figure 2c (flow merge) can be thought as chaff added to flow f_0 and f_0^1, respectively. On the other hand, flow splitting can be thought as a form of packet dropping (or de-chaffing) from the subflow's point of view.

Since these flow transformations would change one flow into a very different flow, many people intuitively believe that these changes would make one flow virtually indistinguishable from other flows. Many existing low-latency anonymous communication systems have used variations of the above flow transformations in addition to any cryptographic operations they may use. For example, Onion Routing [24] uses packet padding, and Tor [9] uses a fixed-size cell which requires repacketization. Both NetCamo [15] and Tarzan [12] deliberately introduce chaff to anonymize the network traffic. Work [19] uses random packet dropping as a means to achieve anonymity in the presence of

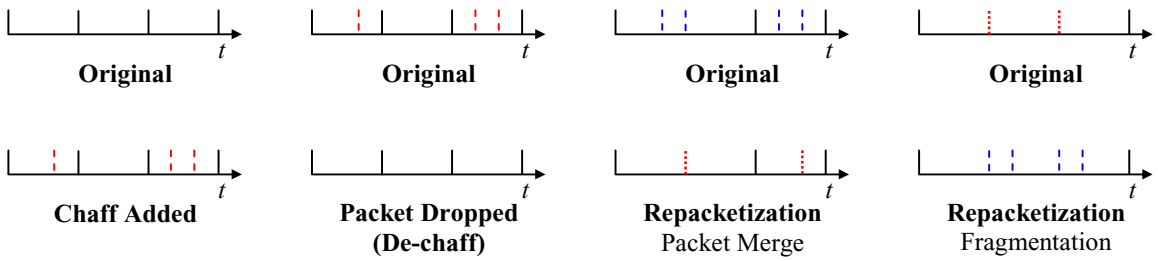

Figure 1. Intra-flow transformations

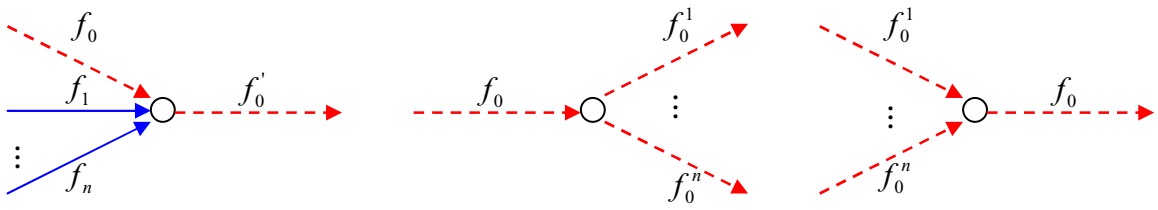

Figure 2. Inter-flow transformations

active timing attacks. Hordes [27] uses multicast, which can be thought as a variation of flow splitting, to provide initiator (or sender) anonymity. All mix based anonymizing systems [4, 22, 17] use some sort of repacketization, packet reordering, or flow mixing to achieve sender anonymity, the receiver anonymity, or the unlinkability of sender and receiver.

Therefore, whether or not we could uniquely identify a network flow despite these flow transformations is a key problem that has a direct impact on some of the very foundations of existing anonymizing techniques. In the rest of this paper, we show that the combination of chaff, packet dropping, repacketization, flow mixing, and flow splitting does not necessarily make one flow indistinguishable from others.

3 Interval Centroid Based Watermarking Scheme

We present a novel watermarking scheme that could make a sufficiently long flow uniquely identifiable even after significant transformations have occurred, such as by adding chaff, packet dropping, flow mixing, and flow splitting/merging. We start with the basic concepts and notions, and then we describe the watermark encoding and decoding processes. We further establish an upper bound on the decoding error probability assuming there is no active countermeasures. We will consider active countermeasures in Section 4. In the rest of this section, we use *packet flow* and *network flow* interchangeably.

3.1 Time Interval and Centroid of Interval

Given a packet flow of duration $T_f > 0$, we want to embed l-bit watermark with redundancy $r > 0$. Starting from offset $o > 0$, we can choose a duration T_d and divide it into $2n$ (where $n = r \times l$) intervals of length T ($T > 0$): $I_0, \ldots, I_{2n-1}$. Assume there are $n_p > 0$ packets $P_1, \ldots, P_{n_p}$ in the $2n$ intervals. Let t_i $(i = 1, \ldots, n_p)$ represent the absolute time stamp of packet P_i, and t_0 be the absolute time stamp of the start point of the first interval. Then $t'_i = t_i - t_0$ is the relative time stamp of P_i from the starting point of the first interval. Apparently, packet P_i would occur within interval $\lfloor t'_i/T \rfloor$.

We are interested in P_i's relative position within its interval, and we use $\triangle t_i$ to represent the P_i's offset from the start point of its interval. Then we have

$$\triangle t_i = t'_i \bmod T \tag{1}$$

Therefore, dividing a duration T_d of a packet flow into equal size intervals is essentially a modulo operation, and the packets' relative positions within their respective intervals are essentially the remainders of the modulo operations on those packets in duration T_d.

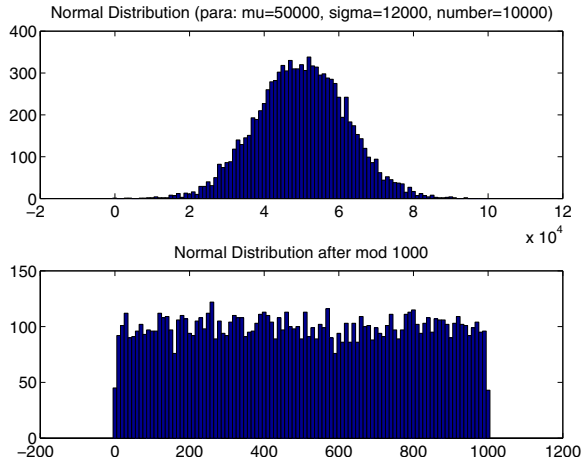

Figure 3. Remainder distribution of modulo operation over normally distributed random variable

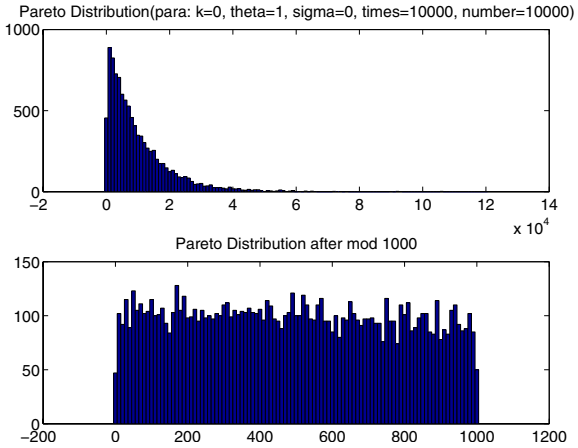

Figure 4. Remainder distribution of modulo operation over exponentially distributed random variable

Given any particular sequence of time stamps $t'_1, \ldots, t'_{n_p}$ and a random interval length $T > 0$, when $T \ll t'_{n_p} - t'_1$ and n are large, $\triangle t_i = t'_i \bmod T$ is approximately uniformly distributed in range [0, T). Figures 3 and 4 show the empirical distributions of the remainders of modulo 1000 operations over normally and exponentially distributed random variables, respectively. They clearly show that the remainder of modulo operation over random variables of different distributions is approximately uniformly distributed.

In other words, given any packet flow with sufficient packets, any randomly chosen offset $o > 0$ and any interval size $T > 0$, the relative positions of all packets within their respective intervals ($\triangle t_i$) are uniformly distributed.

Therefore, the expected value of $\triangle t_i$ is

$$E(\triangle t_i) = \frac{T}{2} \quad (i = 1, \ldots, n) \tag{2}$$

and the variance of $\triangle t_i$ is

$$\text{Var}(\triangle t_i) = \frac{T^2}{12} \quad (i = 1, \ldots, n) \tag{3}$$

Assuming interval I_i $(i = 0, \ldots, 2n-1)$ has $n_i > 0$ packets $P_{i_0}, \ldots, P_{i_{n_i-1}}$, we are interested in the "balance point" of those packets in each interval I_i. We define the *centroid* of interval I_i $(i = 0, \ldots, 2n-1)$ as

$$\text{Cent}(I_i) = \frac{1}{n_i} \sum_{j=0}^{n_i-1} \triangle t_{i_j} \tag{4}$$

In case interval I_i is empty, we define Cent(I_i) to be $\frac{T}{2}$.

3.2 Random Grouping and Assignment of Intervals

We use the following process to independently and randomly choose n intervals out of the $2n$ intervals: we sequentially scan each of the $2n$ intervals and we independently and randomly choose the current interval with probability 0.5. We can expect to have n intervals randomly chosen. We call the n chosen intervals group A intervals and denote them as I_k^A $(k = 0, \ldots, n-1)$. We call the rest of the n intervals group B intervals and denote them as I_k^B $(k = 0, \ldots, n-1)$. Figure 5 shows the random grouping of the time intervals of a packet flow. Apparently, there are $\frac{(2n)!}{n!n!}$ such equal groupings.

Now we randomly determine which intervals (I_k^A and I_k^B, $k = 0, \ldots, n-1$) will be used for encoding watermark bit i $(i = 0, \ldots, l-1)$. We scan each of I_k^A $(k = 0, \ldots, n-1)$ and randomly assign, with probability $\frac{1}{l}$, the current interval for encoding watermark bit i $(i = 0, \ldots, l-1)$. Then we can expect to have $r = \frac{n}{l}$ group A intervals assigned for each watermark bit. We denote the j-th $(j = 0, \ldots, r-1)$ group A interval assigned for watermark bit i $(i = 0, \ldots, l-1)$ as $I_{i,j}^A$. Similarly, we randomly assign each I_k^B $(k = 0, \ldots, n-1)$ for encoding watermark bit i $(i = 0, \ldots, l-1)$, and we denote the j-th $(j = 0, \ldots, r-1)$ group B interval assigned for watermark bit i $(i = 0, \ldots, l-1)$ as $I_{i,j}^B$. Figure 6 illustrates the random assignment of the time intervals for embedding different watermark bits. There are totally $\frac{n!}{(r!)^l}$ different such assignments.

We use $N_{i,j}^A$ and $N_{i,j}^B$ to represent the total packet numbers in interval $I_{i,j}^A$ and $I_{i,j}^B$, respectively. Let

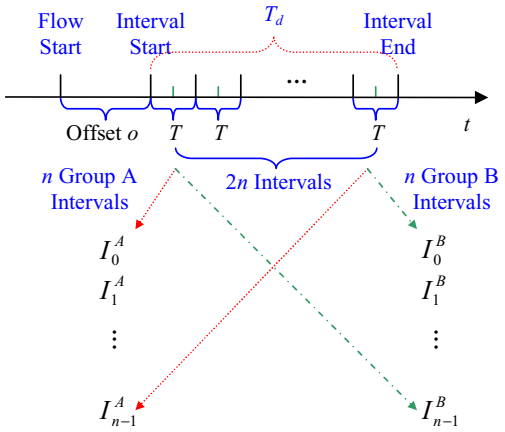

Figure 5. Random grouping of time intervals of packet flow

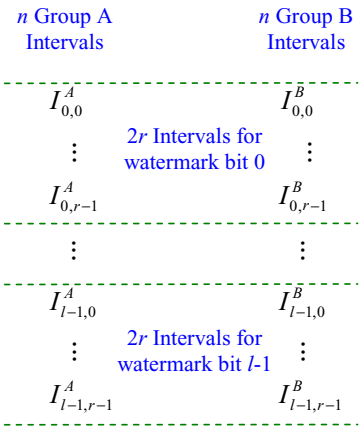

Figure 6. Random assignment of time intervals of packet flow

$$N_i^A = \sum_{j=0}^{r-1} N_{i,j}^A \text{ and } N_i^B = \sum_{j=0}^{r-1} N_{i,j}^B$$

Then N_i^A and N_i^B represent the total packet number of group A and B intervals, respectively, assigned for encoding watermark bit i.

Since we randomly assign each interval, with equal probability, to group A and B, and we randomly assign each group A and B intervals, with equal probability, for encoding each watermark bit, each of the $2n$ intervals has equal probability to be assigned for each watermark bit. In particular, each interval has $\frac{r}{2n} = \frac{1}{2l}$ probability to be assigned for encoding watermark bit i as one of the $I_{i,j}^A$ $(j = 0, \ldots, r-1)$, and each interval has $\frac{r}{2n} = \frac{1}{2l}$ probability to be assigned for encoding watermark bit i as one of the $I_{i,j}^B$ $(j = 0, \ldots, r-1)$. In addition, each of $I_{i,j}^A$ and each of $I_{i,j}^B$ have equal probability to have each of the n_p packets. Therefore, the expected numbers of packets in group A and B intervals for encoding watermark bit i are

$$E(N_i^A) = E(N_i^B) = \frac{n_p}{2l} \quad (5)$$

3.3 Watermark Encoding and Decoding

In this section, we present the encoding and decoding processes of our interval centroid based watermarking scheme, and we formally establish an upper bound on the watermark decoding error probability.

Given a packet flow, offset $o > 0$ and interval size T, we can have $2r$ intervals $I_{i,j}^A$ and $I_{i,j}^B$ $(j = 0, \ldots, r-1)$ randomly grouped and assigned to encode watermark bit i $(i = 0, \ldots, l-1)$. To encode or decode a watermark bit, we aggregate all of the time stamps in the r group A and group B intervals ($I_{i,j}^A$ and $I_{i,j}^B$), respectively, and we calculate the centroids of those packets.

Let

$$A_i = \frac{\sum_{j=0}^{r-1}[N_{i,j}^A \text{Cent}(I_{i,j}^A)]}{\sum_{j=0}^{r-1} N_{i,j}^A} = \frac{\sum_{j=0}^{r-1}\sum_{k=0}^{N_{i,j}^A-1} \triangle t_{i,j,k}^A}{N_i^A} \quad (6)$$

and

$$B_i = \frac{\sum_{j=0}^{r-1}[N_{i,j}^B \text{Cent}(I_{i,j}^B)]}{\sum_{j=0}^{r-1} N_{i,j}^B} = \frac{\sum_{j=0}^{r-1}\sum_{k=0}^{N_{i,j}^B-1} \triangle t_{i,j,k}^B}{N_i^B} \quad (7)$$

where $\triangle t_{i,j,k}^A$ and $\triangle t_{i,j,k}^B$ represent the k-th packet in interval $I_{i,j}^A$ and $I_{i,j}^B$, respectively.

Here, A_i and B_i are aggregated centroids of group A and B packets, respectively, assigned for encoding and decoding watermark bit i, and they are actually the sample means of those N_i^A $\triangle t_{i,j,k}^A$s and N_i^B $\triangle t_{i,j,k}^B$s, respectively, that fall within those r group A and B intervals $I_{i,j}^A$ and $I_{i,j}^B$ $(j = 0, \ldots, r-1)$.

Since $E(\triangle t_{i,j,k}^A) = E(\triangle t_{i,j,k}^B) = \frac{T}{2}$, we have

$$E(A_i) = E(B_i) = \frac{T}{2} \quad (8)$$

We encode and decode watermark bit i into the difference between A_i and B_i. Let

$$Y_i = A_i - B_i \quad (9)$$

then $E(Y_i) = 0$ and Y_i is symmetric around zero.

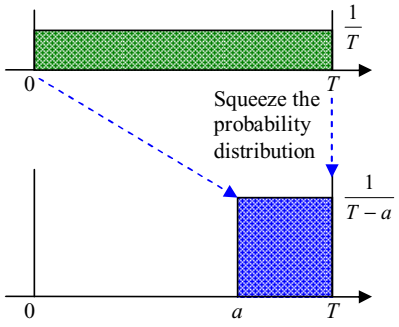

Figure 7. Probability distribution of packet timing before and after watermark encoding delay

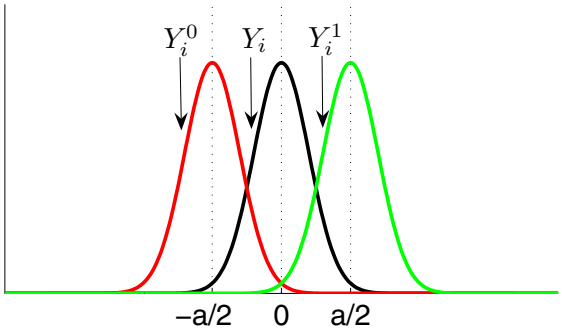

Figure 8. encoding watermark bit by shifting the distribution of Y_i

3.4 Watermark Encoding

To encode bit '1', we deliberately increase A_i so that Y_i will be more likely to be positive than negative. Similarly, to encode bit '0', we deliberately increase B_i so that Y_i will be more likely to be negative than positive.

To increase A_i or B_i, we can simply delay each packet within each interval $I_{i,j}^A$ or $I_{i,j}^B$. Let $0 < a < T$ be the maximum delay to be applied, $\triangle t_{i,j,k}$ be packet $P_{i,j,k}$'s offset from the start of its interval $I_{i,j}$, and $\triangle t'_{i,j,k}$ be the resulting offset after $P_{i,j,k}$ has been delayed. We delay packet $P_{i,j,k}$ according to the following strategy

$$\triangle t'_{i,j,k} \;=\; a + \frac{(T-a)\triangle t_{i,j,k}}{T} \qquad (10)$$

Since $\triangle t_{i,j,k}$ is uniformly distributed on range $[0, T)$, $\triangle t'_{i,j,k} \in [a, T)$. In fact, $\triangle t'_{i,j,k}$ is uniformly distributed[1] on range $[a, T)$. In other words, our delay strategy actually "squeezes" the original uniform distribution of $\triangle t_{i,j,k}$ from range $[0, T)$ to range $[a, T)$. Figure 7 illustrates the effect of our packet delay strategy over the distribution of packets within an interval of size T.

Let A'_i and B'_i be the random variables that denote the resulting values of A_i and B_i, respectively, after all the packets in $I_{i,j}^A$ and $I_{i,j}^B$ $(j = 0, \ldots, r-1)$ have been delayed according to equation 10. Then we have

$$E(A'_i) \;=\; E(B'_i) \;=\; \frac{T+a}{2} \qquad (11)$$

We use $Y_i^1 = A'_i - B_i$ to represent the resulting value of Y_i after bit '1' is encoded by increasing A_i, and we use $Y_i^0 = A_i - B'_i$ to represent the resulting value of Y_i after bit '0' is encoded by increasing B_i. We have

$$E(Y_i^1) \;=\; \frac{a}{2} \quad \text{and} \quad E(Y_i^0) \;=\; -\frac{a}{2} \qquad (12)$$

[1]The derivation of $\triangle t'_{i,j,k}$'s distribution can be found in Appendix A.

where $N_i = \min(N_i^A, N_i^B)$.

Therefore, our watermark encoding actually shifts the distribution of Y_i to the left or right for $\frac{a}{2}$, and it makes the resulting distributions of Y_i^1 and Y_i^0 slightly more clustered than that of Y_i. Figure 8 illustrates the effect of our watermark encoding over distribution of Y_i.

3.5 Watermark Decoding

To decode the watermark from a flow, we can calculate each Y_i $(i = 0, \ldots, l-1)$ given the correct decoding offset o, interval size T, and the exact interval grouping and assignment information. If Y_i is greater than 0, the decoding of watermark bit i is 1; otherwise, the decoding is 0.

Therefore, the probability that encoded bit '0' is mistakenly decoded as bit '1' is $\Pr[Y_i^0 > 0]$, and the probability that encoded bit '1' is mistakenly decoded as bit '0' is $\Pr[Y_i^1 < 0]$.

Now we apply Chebyshev inequality to Y_i^0 and Y_i^1 to derive the upper bound of the decoding error probability.

$$\Pr[|Y_i^1 - E(Y_i^1)| \geq \frac{a}{2}] \;\leq\; \frac{4\mathrm{Var}(Y_i^1)}{a^2} \;\leq\; \frac{T^2 + (T-a)^2}{3a^2 N_i} \qquad (13)$$

Since distribution of $\Pr[Y_i^1 - \frac{a}{2}]$ is symmetric

$$\Pr[Y_i^1 < 0] \;=\; \frac{1}{2}\Pr[|Y_i^1 - E(Y_i^1)| \geq \frac{a}{2}] \;\leq\; \frac{T^2 + (T-a)^2}{6a^2 N_i} \qquad (14)$$

Similarly, we have

$$\Pr[Y_i^0 > 0] \;\leq\; \frac{T^2 + (T-a)^2}{6a^2 N_i} \qquad (15)$$

Therefore, given any T, $0 < a < T$, we can always minimize the decoding error to arbitrarily low by increasing N_i,

which can be achieved by increasing the redundancy number r provided that the flow is long enough with sufficient packets.

4 Properties of the Interval Centroid Based Watermarking Scheme

In this section, we present and analyze a few key properties of the interval centroid-based watermarking scheme that are fundamental to the capability of uniquely identifying a flow even after significant transformation has occurred.

Here, we use a pseudo random number generator (RNG) and a seed s to randomly group and assign each interval to a different watermarking bit. In other words, the random interval grouping and assignment are determined and represented by the RNG and seed s used. Tuple $< o, T, \text{RNG}, s >$ represents the complete information needed for the watermarking encoder and decoder to determine the pseudo-random interval grouping and assignment for encoding and decoding the watermark, and it will be shared only between the watermarking encoder and decoder.

4.1 Self-Synchronization

Being able to self-synchronize during the decoding process is one of the most distinct features of our interval centroid based method compared with all watermarking methods. This property enables our watermarking method to uniquely identify flows even after they have been repacketized, merged/mixed with other flows, split into multiple subflows, and perturbed in packet timing.

Given the watermark embedding parameter tuple $< o, T,$ RNG, $s >$, the decoder should be able to derive the exact random interval grouping and assignment used for encoding the watermark. However, the correct decoding offset not only depends on the value of o but also the clock setting of the decoding host as well as any timing perturbation on the packet timing. When the clock of the watermark decoding host is perfectly synchronized with the clock of the watermark encoding host and there is not packet timing perturbation, offset o will point to the correct decoding start time. When the clocks of the watermark encoding and decoding hosts are not perfectly synchronized, o will point to the wrong decoding start time. In addition, any network delay, network delay jitter, deliberate timing perturbation would shift the correct decoding offset.

Fortunately, our interval centroid-based watermarking could self-synchronize the decoding offset with the encoding offset even if 1) the clocks of the watermark encoding host and decoding host are not synchronized; 2) there is substantial network delay, delay jitter or timing perturbation on the watermarked flows. Due to the symmetric nature

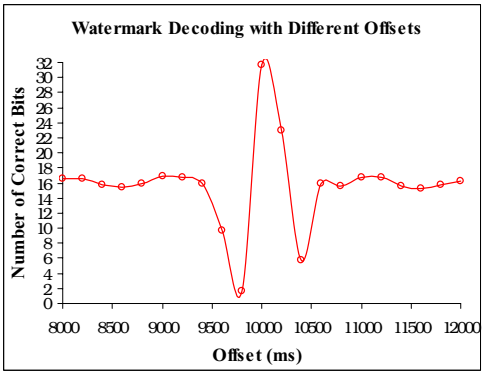

Figure 9. Decoding with different offsets

of our random interval grouping and assignment for watermark encoding and decoding, the decoding with wrong offsets appears random and it tends to have $\frac{l}{2}$ different bits than the encoded l-bit watermark. This property gives us an easy way to determine if the decoding offset used is correct. When decoding from a host without precise clock synchronization with the watermark encoding host, we can simply try a range of different offsets, and the offset that results in the closest match with the watermark is the correct offset for decoding. Figure 9 shows the offline decoding of a flow that was watermarked with a 32-bit watermark (with 10 second offiset) under different offsets, and it clearly shows that only the offset that is very close to the correct one yields the best watermark decoding.

One cost of trying different offsets is that it tends to increase the false-positive rate of the watermark decoding. For example, if we try 10 different offsets for decoding, the resulting watermark decoding false-positive rate could be increased to up to 10 times the original false-positive rate. Nevertheless, we can lower the aggregated false-positive rate of the multi-offset decoding sufficiently by lowering the false-positive rate of the single-offset decoding if we have enough packets.

4.2 Robustness Against Chaff and Flow Mixing

Robustness against chaff and flow mixing/merging is the most important property of our interval centroid-based watermarking scheme. In this subsection, we analyze the strength of our method by establishing an upper bound on the decoding error probability in the presence of chaff.

Given any packet flow of n packets, we consider any other packets added to or mixed with the original flow as *chaff* or *chaff packets*. Assume there are totally m chaff packets $P_{c,1}, \ldots, P_{c,m}$ added to packet flow $P_1, \ldots, P_n$, then the resulting flow $P'_1, \ldots, P'_{m+n}$ (P'_i is either P_j or $P_{c,j}$) is a mix of the original flow and the chaff.

Here we consider all the chaff packets as originating from another random packet flow. From Section 3.1, we know that the relative offsets of all chaff packets within their respective intervals are uniformly distributed. Therefore, the chaff added to a watermarked flow tends to shift the centroid within each interval toward the center of the interval, which would weaken the strength of the embedded watermark. In the rest of this subsection, we quantitatively analyze the negative impact of chaff.

We use $\triangle\hat{t}^A_{i,j,k}$ and $\triangle\hat{t}^B_{i,j,k}$ to represent the offsets of the k-th chaff packet added to the j-th group A interval $I^A_{i,j}$ and group B interval $I^B_{i,j}$, respectively. Let $M^A_{i,j}$ and $M^B_{i,j}$ be the number of chaff packets added to interval $I^A_{i,j}$ and $I^B_{i,j}$, respectively. The $M^A_i = \sum_{j=0}^{r-1} M^A_{i,j}$ and $M^B_i = \sum_{j=0}^{r-1} M^B_{i,j}$ are the total number of chaff packets added to those r group A intervals $I^A_{i,j}$ and r group B intervals $I^A_{i,j}$, respectively, assigned for watermark bit i.

We first consider the statistical characteristics of the offsets of the chaff packets within their intervals. Let

$$C^A_i = \frac{\sum_{j=0}^{r-1}\sum_{k=0}^{M^A_{i,j}-1} \triangle\hat{t}^A_{i,j,k}}{M^A_i} \quad (16)$$

and

$$C^B_i = \frac{\sum_{j=0}^{r-1}\sum_{k=0}^{M^B_{i,j}-1} \triangle\hat{t}^B_{i,j,k}}{M^B_i} \quad (17)$$

Now we consider the impact of the chaff packets over the watermark detection error probability $\Pr[Y^0_i > 0]$ and $\Pr[Y^1_i < 0]$. Let $\hat{Y}^0_i$ and $\hat{Y}^1_i$ be the random variables that denote the resulting values of Y^0_i and Y^1_i, respectively, after chaff has been added. Then the probability that bit '1' is mistakenly decoded as bit '0' is $\Pr[\hat{Y}^1_i < 0]$, and the probability that bit '0' is mistakenly decoded as bit '1' is $\Pr[\hat{Y}^0_i > 0]$. By applying the Chebyshev inequality to $\hat{Y}^0_i$ and $\hat{Y}^1_i$, we establish the following upper bounds on the decoding error probabilities (we omit the detailed derivation due to space limitation):

$$\begin{aligned}\Pr[\hat{Y}^0_i > 0] &= \Pr[\hat{Y}^0_i - E(\hat{Y}^0_i) > -E(\hat{Y}^0_i)] \quad (18)\\ &= \frac{1}{2}\Pr[|\hat{Y}^0_i - E(\hat{Y}^0_i)| \geq -E(\hat{Y}^0_i)]\\ &\leq \frac{\mathrm{Var}(\hat{Y}^0_i)}{2(E(\hat{Y}^0_i))^2} = \frac{(1+R_B)^2T^2}{3a^2N_i(1+R)}\end{aligned}$$

$$\Pr[\hat{Y}^1_i < 0] \leq \frac{\mathrm{Var}(\hat{Y}^1_i)}{2(E(\hat{Y}^1_i))^2} = \frac{(1+R_A)^2T^2}{3a^2N_i(1+R)} \quad (19)$$

Here, $R_A = \frac{M^A_i}{N^A_i}, R_B = \frac{M^B_i}{N^B_i}, R = \frac{M_i}{N_i}$, and they represent the ratios between the number of chaff packets and the number of original packets. By the law of large numbers, $R_A \approx R_B \approx R$ when N_i is large. Equations 18 and 19 show that the larger the R_A, R_B, R, the higher the decoding error probabilities. This result confirms our intuition: the more chaff, the more errors the decoding tends to have. However, no matter how large the R_A, R_B, R (as long as they are finite), we can always make the decoding error probabilities arbitrarily close to zero by having sufficiently large N_i. From equation 5, we can make N_i sufficiently large by having sufficiently large n_p provided the flow is sufficiently long and there are enough packets.

The important result here is that our interval centroid-based watermarking scheme can achieve asymptotic error-free decoding even if the number of chaff packets added is many times more than the number of original packets, provided the original flow is long enough and has enough packets. This result holds true regardless of the distribution of chaff added and it counters the claims by Blum *et al.* [5].

4.3 Robustness Against Packet Dropping, Repacketization, and Flow Splitting

Packet dropping, merging adjacent packets, and splitting flows into multiple subflows will decrease the number of packets in the original flow. We can summarize this effect as de-chaff. Since we randomly divide flow duration into multiple intervals and randomly group those intervals for each watermark bit, any packet lost should be uniformly distributed within each interval $I^A_{i,j}$ and $I^B_{i,j}$. Therefore, when there are enough packets left in the flow, the centroids of all the intervals tend to remain the same even after the packets have been randomly dropped and merged. This property would allow the embedded watermark to persist even after random packet dropping, merging, or flow splitting.

5 Experiments

5.1 Real-Time Experiments on Live Anonymized Web Traffic

We conducted real-time penetration experiments on the Total Net Shield service provided by leading anonymizing service provider www.anonymizer.com. According to www.anonymizer.com, the Total Net Shield is their "ultimate solution in online identity protection". This makes it an attractive candidate for examining the effectiveness of our flow watermarking technique in identifying transformed network flows.

Figure 10 illustrates the setup of our real-time experiments. We watermarked the live Web traffic from www.usatoday.com and we wanted to see if our watermark could penetrate the the Total Net Shield service provided by www.anonymizer.com. We set up an Apache server

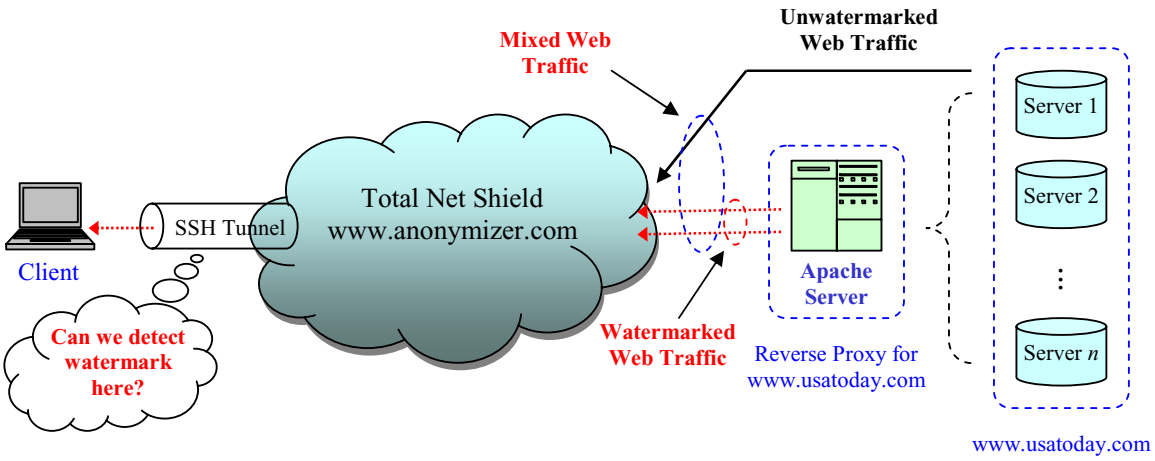

Figure 10. Experimental setup for penetrating the Total Net Shield service by www.anonymizer.com

as a reverse proxy to www.usatoday.com so that we could access all the Web pages of www.usatoday.com by pointing a browser to the URLs of our Apache server. This setup also enabled us to watermark most Web traffic from www.usatoday.com at our Apache server machine. We noticed that our Apache reverse proxy did not catch all of the Web traffic between the Web servers of www.usatoday.com and our client. In a separate experiment, we found that about 11% of the Web traffic from www.usatoday.com, most of which was related to JavaScript, did not pass our Apache server. Therefore, when we watermarked the Web traffic from www.usatoday.com at our Apache server machine, we only watermarked 89% of them. In other words, when the Web traffic from www.usatoday.com reached the entry point(s) of anonymizer.com, it consisted of watermarked traffic mixed with unwatermarked traffic.

From the client machine, we browsed various Web pages of www.usatoday.com through anonymizer.com's Total Net Shield. All of the HTTP traffic between the client machine and anonymizer.com was transferred through the SSH tunnel. From our Apache server's point of view, all of the HTTP requests originated from hosts within anonymizer.com. We also noticed that one click from the client's browser could trigger multiple connections between our Apache server and multiple access points of anonymizer.com.

We conducted our experiments between 10:00am to 20:00pm from April 19 to 24, 2006. We chose to use a 500 ms time interval, 350 ms maximum delay, and 10 seconds offset for embedding 32-bit watermarks into the live Web traffic. We randomly generated 100 32-bit watermarks, each of which had a Hamming distance of at least 12 to any other watermarks. We used redundancy numbers 12, 14, 16, 18, and 20, which required the Web traffic duration from 394 to 650 seconds. For each redundancy number, we conducted 20 separate experiments with 20 different watermarks. Table 1 shows the statistics of the Web traffic collected at the client machine and the Apache server machine. For all redundancy numbers, both the average packet numbers and average packet rates at the client side were about 90% of those at the server side. However, the average packet sizes at the client side were only about 43% of the average packet sizes at the server side. As a result, the average information flow rates (in terms of bytes/second) at the client side were around 39% of those at the server side. This observation indicates that anonymizer.com had removed over 60% information from the Web traffic returned by www.usatoday.com. Such a drastic content filtering has made the inter-packet timing characteristics of the flows before and after the anonymizer's Total Net Shield appear completely different.

Despite of the significant flow transformations (i.e., repacketization, flow mixing, and packet dropping) and network delay jitter introduced by www.anonymizer.com to the Web traffic, we were able to achieve surprisingly good results in linking the information sender and receiver through our flow watermarking technique. When we decoded the 32-bit watermark from a network flow, we allowed a few bits mismatched with the watermark we were seeking. The number of allowed mismatched bits is called the Hamming distance threshold in our watermark decoding. Figure 11 shows that we can achieve a 100% watermark detection rate with a Hamming distance thresholds of 5, 6, 7, and 8, respectively, and redundancy of 20 from the Web traffic received at the client side. This only requires less than 11 minutes active browsing. With less than $6\frac{1}{2}$ minutes of active browsing traffic, we were able to achieve a 60% watermark detection rate with a Hamming distance threshold of 5. By decoding each of the 100 watermarked network flows with 99 different watermarks, we calculated the watermark de-

Redundancy Number		12	14	16	18	20
Average Number of Packets	from Server	3748	4301	4745	5246	5489
	to Client	3356	3884	4332	4774	5031
Packet Number Ratio		89.54%	90.30%	91.29%	91.00%	91.66%
Average Packet Rate (Pkt/sec)	from Server	8.83	8.90	8.83	8.57	8.19
	to Client	7.88	8.02	8.05	7.79	7.54
Packet Rate Ratio		89.24%	90.11%	91.17%	90.90%	92.06%
Average Packet Size (Byte)	from Server	801.84	806.45	804.52	812.73	814.09
	to Client	354.28	349.18	351.89	346.80	353.85
Packet Size Ratio		44.18%	43.30%	43.74%	42.67%	43.47%
Average Information Flow Rate (Byte/sec)	from Server	7080.23	7177.37	7103.87	6965.11	6667.36
	to Client	2791.69	2800.45	2832.69	2701.56	2668.04
Information Flow Rate Ratio		39.43%	39.02%	39.88%	38.79%	40.02%

Table 1. Packet Flow Statistics of Online Experiments

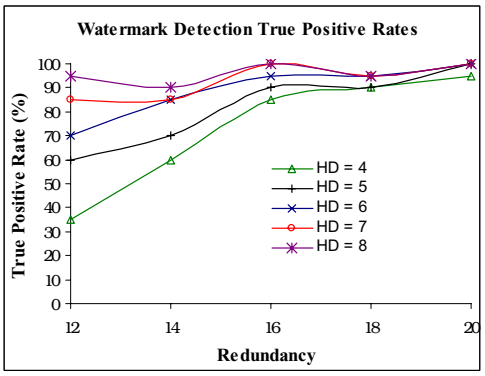

Figure 11. Watermark detection true positive rate with different redundancy numbers and different Hamming distance thresholds

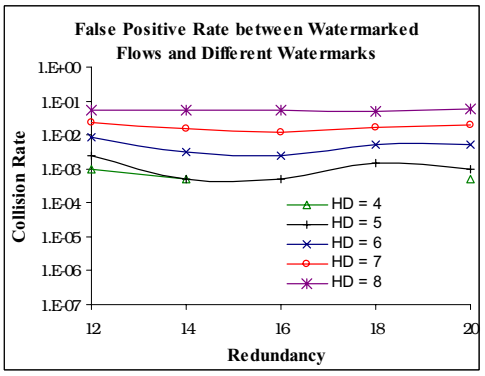

Figure 12. Watermark detection false positive rate with different redundancy numbers and different Hamming distance thresholds

tection false positive rate. Figure 12 shows that we achieved a less than 0.3% watermark detection false positive rate with a Hamming distance threshold of 5 for all redundancy levels. These results on live Web traffic confirmed that our flow watermarking technique can effectively penetrate the anonymizing service provided by www.anonymous.com.

5.2 Offline Experiments

We obtained 100 watermarked flows by encoding a synthetic flow, generated by tcplib [8], with 100 different 32-bit watermarks. We used redundancy number $r = 20$, interval size $T = 500$ms, and timing adjustment $a = 350ms$. For each watermarked flow, we further generated 10 transformed flows by adding uniformly distributed chaff (or bogus packets) of rates between 10 to 100 packets/second. We then tried to decode the 1000 transformed watermarked flows with correct watermarking parameters. For all rates of chaff, we are able to obtain a 100% watermark detection rate while getting no more than a 0.5% watermark detection false positive rate. Since the average packet rate of the original unwatermark synthetic flow is only 0.8 packet/second, the flow we watermarked has only about 512 packets. A chaff rate of 100 packets/second is over 120 times more than the packet rate of the original flow. We also tried watermark decoding with normally distributed chaff, and we could achieve virtually a 100% watermark detection rate when the normally distributed chaff is 4 times more than the original packets.

These results confirmed our claim that it is possible to uniquely identify a long network flow even if the amount of chaff added to the flow is many times more than the number of original packets as indicated by equations (18) and (19). This also implies that a flow can be uniquely identified even if it is mixed with other flows. Our offline ex-

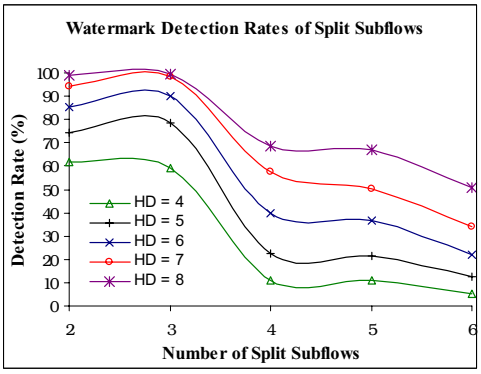

Figure 13. Watermark detection rates of split Subflows

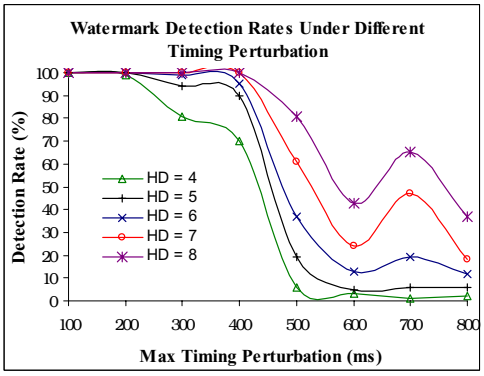

Figure 14. Watermark detection rates with timing perturbation

periments also confirmed that when two watermarked flows are mixed, both watermarks could be successfully decoded from the mixed flow by our interval centroid based watermarking scheme.

For each of the 100 watermarked flows (with 100 different watermarks), we randomly and uniformly split them into 2, 3, 4, 5, 6 subflows. We tried to detect the watermark from each of the subflows. Figure 13 shows the average watermark detection rates from various subflows under different Hamming distance thresholds. The figure indicates that splitting a flow into a few subflows does not make a flow unidentifiable as long as each subflow has a reasonable number of packets. In specific, we could get about 80% watermark detection rate with Hamming Distance threshold 5 on each of the 3 subflows that were split from a 512-packet flow. These results imply that a watermarked flow can be uniquely identified even if substantial portion of its packets have been dropped.

To evaluate the robustness of our interval centroid based watermarking scheme against timing perturbation, we introduced uniformly distributed random timing perturbation to every packet of the 100 watermarked flows, and tried to detect the watermark from the perturbed flows. Figure 14 shows the average watermark detection rates under various levels of random timing perturbation. It clearly indicates that the interval centroid based watermarking scheme is robust against any timing perturbation that is less than the interval size T.

5.3 Discussion

Our experimental results confirm that our interval centroid based watermarking scheme is highly effective in identifying sufficiently long flows even after significant transformations have occurred. This technique allows us to effectively link an anonymized packet flow to the original packet flow. Here the number of packets in a packet flow is the fundamental limiting factor of the robustness of our flow watermarking scheme against various flow transformations. As shown in our real-time experiments on www.anonymizer.com, our flow watermarking technique only need about 5,500 packets to penetrate the best anonymizing service of www.anonymizer.com, which uses combinations of flow mixing, repacketization and substantial packet dropping in addition to timing perturbation due to network delay jitter. For single flow transformation such as adding cover traffic, packet dropping, flow mixing, flow splitting, flow merging, our flow watermarking technique could be effective on packet flows of only a few hundred packets.

Since we can watermark a network flow from its source (e.g. from the Web site), we can make sure that whenever a flow is watermarked, only the watermarked flow is observable by others. Without access to the original unwatermarked flow, it would be very difficult, if possible at all, for any one to tell whether an arbitrary flow has been watermarked. In addition, our random grouping and assignment of intervals would make it difficult to detect the existence of watermark in a flow via statistical analysis of the inter-packet timing characteristics.

6 Related Works

A number of low-latency anonymous communication systems [1, 24, 9, 25, 12, 15, 22, 4, 17] have been developed based on proxies or MIXes. Notably, Onion Routing [24], and its second generation, Tor [9], use public key encryption on a pre-determined sequence of proxies to protect the transport of TCP flows. Crowds [25] uses randomly selected proxies to hide the information's sender and receiver. However, none of these methods were designed to provide the unlinkability of sender and receiver. NetCamo [15] and

Tarzan [12] used cover traffic to provide low-latency anonymous communication. A leading anonymous communication service provider, www.anonymizer.com [1], uses multiple proxies and a number of flow transformations (i.e., repacketization, packet dropping, flow mixing/merging) to provide its low-latency anonymous communication services. Instead of relying on proxies or MIXes, Hordes [27] leverages multicasting to provide sender anonymity. P5 [26] uses broadcast to provide sender-, receiver-, and sender-receiver anonymity assuming the adversary is passive.

There are also substantial works on attacking the privacy of Web application and low-latency anonymous communication systems. Felton and Schneider [11] identified an exploit of the Web cache which would allow a malicious Web site to infer whether its visitors have visited some other Web pages. Sun *et al.* [28] investigated how to statistically identify Web pages based on HTTP object size. Levine *et al.* [19] investigated passive timing-based attacks on low-latency anonymizing systems with the assumption that the attacker could control both the first and the last mix in the anonymizing network. However, they only provided a theoretical analysis on general low-latency anonymizing models, and no real experiments or tests were performed on real systems. Murdoch *et al.* [20] proposed a low-cost timing attack on Tor [9] with the assumption that the attacker could control a corrupt Tor node. Wang *et al.* [29] proposed an active watermarking technique that can uniquely identify VoIP flows anonymized by www.findnot.com [2]. Compared with their watermarking method, ours is able to self-synchronize during the watermark decoding process which makes our watermarking scheme robust against such flow transformations as repacketization, packet dropping, flow mixing/merging/splitting in addition to timing perturbation.

Fu *et al.* [13] analyzed the effectiveness of traffic padding in resisting traffic analysis and showed that constant rate traffic padding is not optimal. Gogolewski *et al.* [14] investigated the implications when a user could only choose from a limited subset of all possible proxies in anonymous communication and showed that the anonymity could be degraded dramatically even if the set of all possible proxies was large. Kesdogan *et al.* [18] investigated the theoretical limits of the anonymity provided by the MIXes in the presence of omnipresent passive adversary. However, their result is limited to the MIXes that do not introduce any bogus traffic or dummy messages. Compared with their analysis, ours does not require the global monitoring capability, and our attack is effective even if the anonymizing system introduces bogus traffic or bogus messages.

Peng *et al.* [21] proposed an offline statistical method to detect the existence of watermark embedded in a network flow by method [30]. However, their watermark detection method assumes the watermark embedding follows some simple patterns and requires access to both the unwatermarked and watermarked flows to be effective. Since our interval centroid based watermarking scheme 1) uses non-trivial random grouping and assignment of intervals and 2) could prevent others from accessing the unwatermarked flow by watermarking a flow from its source, it is unlikely that method [21] could detect our watermark from a given flow offline.

7 Conclusions

It is a common belief that drastic flow transformations would effectively disguise network flows and traffic padding is effective in anonymizing network flows. The main contribution of this paper is that we have demonstrated that existing flow transformations, such as adding bogus packet, packet dropping, flow mixing, flow splitting, and flow merging, in addition to timing perturbation do not necessarily make a long network flow indistinguishable from others.

By developing a novel flow watermarking technique, we are able to uniquely identify a long flow even if it has gone through drastic flow transformations. In particular, our analysis has revealed a rather surprising result regarding the inherent limitation of flow transformations in anonymizing network flows – a sufficiently long network flow could be uniquely identified even if the amount of cover traffic (or bogus packets) is many times more than the original packets. In addition to demonstrating the theoretical limitations of the flow transformation based low-latency anonymizing systems, we have also developed the first practical attack on a leading commercial anonymizing system. Our real-time experiments on www.anonymizer.com have shown that we only need a little over 10 minutes of active surfing traffic (about 5500 packets) to penetrate the Total Net Shield service provided by www.anonymizer.com.

Our packet flow watermarking attack is based on the packet timing correlation between the original packet flow and the anonymized packet flow. Since no practical low-latency anonymizing system could remove all the mutual information from the packet timing domain, our flow watermarking attack is applicable to all practical low-latency anonymous communication systems. It is an open research problem to determine whether and to what extent we can practically achieve good anonymity through low-latency anonymous communication systems in the presence of active adversary.

Acknowledgement The authors would like to thank the anonymous reviewers for their insightful comments that helped to improve the presentation of this paper. This work was partially supported by NSF Grants CNS-0524286 and CT-0627493.

References

[1] Anonymizer Inc. http://anonymizer.com.

[2] Findnot. http://findnot.com.

[3] O. Berthold, H. Federrath, and M. Kohntopp. Project anonymity and unobservability in the internet. In *Proceedings of Computers Freedom and Privacy*, April 2000.

[4] O. Berthold, H. Federrath, and S. Köpsell. Web MIXes: A system for anonymous and unobservable Internet access. In H. Federrath, editor, *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 115–129. Springer-Verlag, LNCS 2009, July 2000.

[5] A. Blum, D. Song, and S. Venkataraman. Detection of interactive stepping stones: Algorithms and confidential bounds. In *Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection (RAID 2004)*, October 2004.

[6] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudo-nyms. *Communications of the ACM*, 4(2), February 1981.

[7] D. Chaum. The dining cryptographers problem: Unconditional sender and receipient untraceability. *Journal of Cryptography*, 1(1), 1988.

[8] P. B. Danzig and S. Jamin. Tcplib: A library of tcp internetwork traffic characteristics. Technical Report USC-CS-91-495, University of Southern California, 1991.

[9] D. Dingledine, N. Mathewson, and P. Syverson. Tor: The second generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, August 2000.

[10] D. Donoho, A. Flesia, U. Shankar, V. Paxson, J. Coit, and S. Staniford. Multiscale stepping stone detection: Detecting pairs of jittered interactive streams by exploiting maximum tolerable delay. In *Proceedings of the 5th International Symposium on Recent Advances in Intrusion Detection (RAID 2002)*, pages 17–35, October 2002.

[11] E. Felton and M. Schneider. Timing attacks on web privacy. In *Proceedings of the 7th ACM Conference on Computer and Communications Security (CCS 2000)*, pages 25–32, November 2000.

[12] M. Freedman and R. Morris. Tarzan: A peer-to-peer anonymizing network layer. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002)*, pages 193–206, 2002.

[13] X. Fu, B. Graham, R. Bettati, and W. Zhao. On countermeasures to traffic analysis attacks. In *Proceedings of the 2003 IEEE Workshop on Information Assurance*, pages 188–195. IEEE, June 2003.

[14] M. Gogolewski, M. Klonowski, and M. Kutylowski. Local view attack on anonymous communication. In *Proceedings of ESORICS 2005*, September 2005.

[15] Y. Guan, X. Fu, D. Xuan, P. U. Shenoy, R. Bettati, and W. Zhao. Netcamo: Camouflaging network traffic for qos-guaranteed mission critical applications. *IEEE Transactions on Systems, Man, and Cybernetics*, 31(4):253–265, July 2001.

[16] Y. Guan, C. Li, D. Xuan, R. Bettati, and W. Zhao. Preventing traffic analysis for real-time communication networks. In *Proceedings of Military Communications Conference (MILCOM 1999)*, pages 744–750. IEEE, November 1999.

[17] A. Jerichow, J. Müller, A. Pfitzmann, B. Pfitzmann, and M. Waidner. Real-Time MIXes: A Bandwidth-Efficient Anonymity Protocol. *IEEE Journal on Selected Areas in Communications*, 16(4), 1998.

[18] D. Kesdogan, D. Agrawal, V. Pham, and D. Rautenbach. Fundamental Limits on the Anonymity Provided by the MIX Technique. In *Proceedings of the 2006 IEEE Symposium on Security and Privacy*, pages 86–99, 2006.

[19] B. Levine, M. Reiter, C. Wang, and M. Wright. Timing attacks in low-latency mix systems. In *Proceedings of 8th Financial Cryptography*, 2004.

[20] S. Murdoch and G. Danezis. Low-cost traffic analysis of tor. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy*, pages 183–195, May 2005.

[21] P. Peng, P. Ning, D. S. Reeves. On the Secrecy of Timing-Based Active Watermarking Trace-Back Techniques. In *Proceedings of the 2006 IEEE Symposium on Security and Privacy*, pages 334-348, 2006.

[22] A. Pfitzmann, B. Pfitzmann, and M. Waidner. ISDN-mixes: Untraceable Communication with Very Small Bandwidth Overhead. In *Proceedings of the GI/ITG Conference on Communication in Distributed Systems*, pages 451–463, February 1991.

[23] A. Pfitzmann and M. Waidner. Networks without user observability - design options. *Computers and Security*, 6(2):158–166, 1987.

[24] M. Reed, P. Syverson, and D. Goldschlag. Anonymous connections and onion routing. *IEEE JSAC Copyright and Privacy Protection*, 1998.

[25] M. K. Reiter and A. D. Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, November 1998.

[26] R. Sherwood, B. Bhattacharjee, and A. Srinivasan. P5: A protocol for scalable anonymous communication. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, May 2002.

[27] C. Shields and B. N. Levine. A protocol for anonymous communication over the internet. In *Proceedings of the 7th ACM Conference on Computer and Communications Security (CCS 2000)*, pages 33–42, 2000.

[28] Q. Sun, D. R. Simon, Y.-M. Wang, W. Russell, V. N. Padmanabhan, and L. Qiu. Statistical identification of encrypted web browsing traffic. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pages 19–30, Berkeley, California, May 2002.

[29] X. Wang, S. Chen, and S. Jajodia. Tracking anonymous peer-to-peer voip calls on the internet. In *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS 2005)*, pages 81–91, Alexandra, VA, November 2005. ACM.

[30] X. Wang and D. Reeves. Robust correlation of encrypted attack traffic through stepping stones by manipulating of interpackets delays. In *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS 2003)*, pages 20–29, 2003.

[31] X. Wang, D. Reeves, and S. Wu. Inter-packet delay based correlation for tracing encrypted connnections through stepping stones. In *Proceedings of the 7th European Symposium on Research in Computer Security (ESORICS 2002)*, LNCS-2502, pages 244–263. Springer-Verlag, 2002.

2007 IEEE Symposium on Security and Privacy(SP'07)
0-7695-2848-1/07 $20.00 © 2007 IEEE

[32] K. Yoda and H. Etoh. Finding a connection chain for tracing intruders. In *Proceedings of the 6th European Symposium on Research in Computer Security (ESORICS 2000)*, October 2000.

[33] Y. Zhang and V. Paxson. Detecting stepping stones. In *Proceedings of the 9th USENIX Security Symposium*, August 2000.

Appendix

A Derivation of Distribution of $\triangle t'_{i,j,k}$

We use X to represent random variable $\triangle t_{i,j,k}$ and use Y to represent random variable $\triangle t'_{i,j,k}$. Let $r(X) = a + \frac{(T-a)X}{T}$ and $s(Y) = \frac{(Y-1)T}{T-a}$, then $Y = r(X)$ and $X = s(Y)$.

Let $f(x)$ be the $p.d.f.$ of random variable $\triangle t_{i,j,k}$, and $g(y)$ be the $p.d.f$ of random variable $\triangle t'_{i,j,k}$. We have $f(x) = \frac{1}{T}$ for $X \in [0, T)$.

Since both $r(X)$ and $s(Y)$ are continuous, strictly increasing and differentiable, we have

$$\begin{aligned} g(y) &= \frac{dG(y)}{dy} = f[s(y)]\frac{ds(y)}{dy} \qquad \text{(A-1)} \\ &= \frac{1}{T} \times \frac{T}{T-a} = \frac{1}{T-a} \end{aligned}$$

Therefore, random variable $\triangle t'_{i,j,k}$ is uniformly distributed on range $[a, T)$.