面向流量测试的网络流水印标记和溯源模型

丁嘉宁[1,2,3]，张鹏[1,2]，杨嵘[1,2]，刘庆云[1,2]，郭莉[1,2]

(1. 中国科学院 信息工程研究所，北京 100093；2. 信息内容安全技术国家工程实验室，北京 100093；

3. 中国科学院大学，北京 100049)

摘　要：为了在测试床中对测试流量进行标记和溯源，提出了一种基于时间间隔的网络流水印模型，当生成测试流量时，该模型首先把水印内容转换成 0-1 比特序列，然后将 0-1 比特序列转换成流中数据包发送的时间间隔从而实现对流量的标记。当接收测试流量时，该模型通过将流中数据包的时间间隔转换成 0-1 比特序列，进而获取对应的水印内容，从而实现测试流量的溯源。理论分析表明，该模型能够抵御多种攻击手段，同时大量实验证明了该模型在不丢包情况和丢包情况下对测试流量进行溯源的有效性。

关键词：测试床；流水印；流溯源；流标记

中图分类号：TP393　　**文献标识码：**A　　**文章编号：**

Network testbed oriented flow watermarking and provenance tracing model

DING Jia-ning[1,2,3], ZHANG Peng[1,2], YANG Rong[1,2], LIU Qing-yun[1,2], GUO Li[1,2]

(1. Institute of Information Engineering, Chinese Academy of Science, Beijing 100093, China;

2. National Engineering Laboratory for Information Security Technologies, Beijing 100093, China;

3. University of Chinese Academy of Sciences, Beijing 100049, China)

Abstract: In order to label and trace the provenance of any network simulation flow in network testbed, an interval-based flow watermarking and provenance tracing model was proposed. When a simulation flow was generated, this model first transformed the user's watermarking content into 0-1 bit sequence and then sent packets of the flow at particular intervals according to the 0-1 bit sequence to label the flow. When the simulation flow was captured by the model, the time intervals between packets in the flow were transformed into the 0-1 bit sequence so that the watermarking content could be extracted to trace the provenance of this simulation flow. The resilience against various known attack techniques is illustrated through theoretical analysis. Moreover, a large number of experiments prove the validity of this model in tracing simulation flows under both normal and abnormal circumstance.

Key words: testbed; flow watermarking; provenance tracing; flow labeling

1　引言

在大型网络系统正式发布之前，通常使用网络测试床对其性能进行全方位的评估。随着云计算的出现，网络测试床需要支持大规模用户进行在线测试，这给网络测试床的性能和功能提出了更多的要求。首先，用户希望实现测试流量的溯源功能，因为在很多测试用例中，测试流量与用户没有建立关联会导致测试床无法获知测试流量的发起者。其次，用户想要将某些信息附着在测试流量中，但是，在数据包中加入这些信息会带来额外开销，不仅如此，若直接将这些信息放在数据包中，一方面在公共网络测试环境下缺乏安全性，另一方面，数据包的空间有限导致可容纳的信息有限。然而，传统的网络测试床(network test bed)[1~6]在设计和实现的时候，重点都放在真实的流量模拟和网络环境控制等方面，很少考虑对测试流量进行标记和溯源以实现上述功能。最近，一种被称作网络流水印(network flow watermark)的技术在多个场景中得到了应用[7~12]，该技术根据流式数

基金项目：国家自然科学基金资助项目(61402464)；中国博士后基金(2013M541076)

Foundation Items: National Natural Science Foundation of China (61402464); China Postdoctoral Science Foundation (2013M541076)

据的特点，利用不同数据包之间时间差值的随机性达到对数据流进行水印标记的效果。这种水印具有良好的隐蔽性，同时不会带来额外的带宽开销。网络流水印的一般模型通常由标记方(watermarker)和溯源方(decoder)组成，其中标记方负责将水印内容标记到给定的数据流中，溯源方从被标记的数据流中将水印内容还原出来，从而实现了水印效果。为了在测试床中对测试流量进行标记和溯源，本文提出了一种在网络测试床中基于时间间隔(interval-based)的网络流水印标记和溯源模型，其中水印内容是由可以表示任意信息的 0-1 比特序列组成，并通过变换水印位置和设置带有冗余的间隔达到安全、稳定传输的目的，主要贡献如下：

- 在流量测试床中，提出了基于时间间隔的网络流水印模型，实现了对测试流量的标记和溯源的功能，其中测试流量可以被标记任意信息。
- 从理论和实验两方面证明了被标记的测试流量在传输过程中能够有效应对数据包丢失、注入，流重放和多流攻击[13]的情况。

文章的内容组织如下：第二节介绍了相关的工作。第三节从三个层面来说明网络流水印模型的设计与构造，并完整描述了基于该模型的流量标记和溯源过程。第四节分析了基于该模型的溯源过程的正确率，并从攻击者的角度对模型的安全性进行了分析。第五节是实验和评价，第六节是全文总结。

2 相关工作

本文的工作主要涉及网络测试床和网络流水印，下面从这两方面介绍目前的相关工作。

PlanetLab[1]和 GENI[2]是为研究和测试人员提供计算和网络资源的分布式网络测试床，它们主要致力于提供最稳定可靠的基础服务，而不会关注于上层的应用。随后，建立在它们基础之上的针对网络应用的测试环境 VINT[6]，Trellis[4]实现了网络基础架构的模拟以及网络环境的模拟，接着在这些测试床的基础之上又出现了一系列的网络模拟和仿真系统，例如 OPNET[3]、Ns-2[5]等，它们在流量生成，控制流量参数，正确性分析、系统性能和协议方法测试方面做了大量的工作，为许多研究和测试提供了极大的便利。但是以上工作均没有涉及到如何实现测试床中的流量标记和溯源，这也是本文工作的动机之一。

网络流水印在跳板链[14]、匿名通信系统[15]，甚至传感器网络领域都有广泛的应用。其中文献[8]提出的一种基于包间隔(packet-based)的水印技术将水印内容直接通过数据流中数据包之间的时间差值来表示，这是一种基于数值的方法，该方法在数据包丢弃或者重排的情况下不具备健壮性。基于时间间隔(interval-based)的水印技术[7][9][12]，通过设置一段时间内数据包的时间序列分布间接地表示水印信息，然而这种方法易遭到多流攻击。为此RAINBOW[10]、SWIRL[12]和文献[11]提出了进一步的改进，但它们可标记的水印内容是位置信息或者固定信息，而且被标记之后的数据流容易暴露水印的存在[17]。此外，针对流水印模型的攻击[13][17][21]致力于能够感知水印存在、探测水印参数、篡改水印内容。此外[11][16]中提到了针对流式数据的重放攻击。

本文在上述工作的基础上，首次提出了在网络测试床中通过网络流水印对测试流量进行标记和溯源，其中水印内容是由可以表示任意信息的0-1比特序列组成，并通过变换水印位置和设置带有冗余的间隔达到安全、稳定传输的目，同时该方法有效保证了大规模用户在线测试时能够对测试流量进行正确溯源。

3 网络流水印模型

本文将一个数据流定义为流中的数据包在一组时间段中的分布：

$$I_i = [o + iT, o + (i+1)T], i = 0,1,..., N \tag{3.1}$$

$$P_i = \{t_1^i, t_2^i, \ldots, t_{CS}^i\} \tag{3.2}$$

其中P_i为I_i内对应的数据包时间分布，称作**包模式**，*CS* 表示数据包的个数，I_i表示一段自然时间段，*T* 表示时间段长度，*o* 可以通过流的开始时间确定。下面从水印内容、内容位置、水印表示三方面介绍网络流水印模型及实现。

3.1 水印内容

本文的网络流水印模型标记的水印内容是具有实际意义的 IP 地址或者状态信息。为了将其转化为可以被标记的水印，该模型利用包模式表示 0-1 比特信息：

$$bit(P_i) = \begin{cases} 0 & , P_i = p_0 \\ 1 & , P_i = p_1 \end{cases} \tag{3.3}$$

小写的p_0、p_1分别表示比特 0 和 1 对应的包模式，这是从溯源方的角度来看的。标记方需要将水印内容的字符串 *S* 表示成 0-1 比特序列的形式：

$$S^{bit} = map(S) = \{b \mid b = 0,1\} \tag{3.4}$$

映射函数 *map(·)*是私有映射。值得注意的是，这里的映射函数 *map(·)*是可逆的，即在 O(1)时间内有：

$$map^{-1}(S^{bit}) = S \tag{3.5}$$

为了确定字符串的长度，在每个水印内容的头部添加字符串长度L_S的二进制表示$H = \{b_1, b_2, \ldots, b_{L_H}\}$，其中$L_H$是固定的**头部长度**，以及一个伪时间戳：

$$PTS = Hash_1(\omega) = \{b_1, b_2, \ldots b_{L_{PTS}}\} \tag{3.6}$$

其值是固定长度L_{PTS}的标记方当前时间 ω 的哈希值，因此 *PTS* 的内容不会重复出现，也不会呈现固定的

递增模式。之后得到水印内容：

$$DS^{raw} = PTS \| H \| S^{bit} \tag{3.7}$$

其中$\|$表示连接算符，则总的**头部信息长度**为：

$$L_{head} = L_{PTS} + L_H \tag{3.8}$$

3.2 位置选择

为了增强安全性，网络流水印模型将DS^{raw}的比特位置变化之后再将其标记，而不是每次都将DS^{raw}按照固定的顺序(例如最常见的就是第i个位置对应第i位比特)标记。由于伪时间戳本身就是决定内容位置的参数，所以该模型对伪时间戳信息PTS不做位置变化，只将H和S^{bit}的部分进行置换，即自然序列顺序：

$$\begin{aligned} P_h^H &\Rightarrow H(h), h = 1,\ldots L_H \\ P_j^S &\Rightarrow S^{bit}(j), j = 1,\ldots L_S \end{aligned} \tag{3.9}$$

通过置换之后：

$$\begin{aligned} P_{\pi(h)}^H &\Rightarrow H(h) \\ P_{\pi(j)}^S &\Rightarrow S^{bit}(j) \end{aligned} \tag{3.10}$$

$$\begin{aligned} \pi(h) &= Hash_2(PTS \| h \| V_H^* \| K_t) \bmod L_H \\ \pi(j) &= Hash_3(PTS \| j \| V_S^* \| K_t) \bmod L_S \end{aligned} \tag{3.11}$$

其中V^*表示H和S^{bit}中比特 1 的数量，它们不会受到位置变化的影响；K_t是安全密钥，只为标记方和溯源方所有。由于PTS会随着标记方的时间而变化，从而不同测试流量的水印内容即使相同，其所在流量中的位置也会不同；$Hash(\cdot)$是一个安全哈希函数，会产生均匀分布的消息摘要实现水印位置的均匀随机变化。需要注意的一点是，DS^{raw}长度不固定，溯源方无法判断水印的结束位置，也无法获得公式(3.11)第二行中的L_S，所以这里使用固定头部长度L_H获得的 H 来表示不固定的内容长度L_S。置换前后的比特序列满足如下关系：

$$\begin{aligned} H^{\pi}(\pi(h)) &= H(h), h = 1,\ldots, L_H \\ S^{\pi}(\pi(j)) &= S^{bit}(j), j = 1,\ldots, L_S \end{aligned} \tag{3.12}$$

最终被标记的水印内容

$$DS^w = PST \| H^{\pi} \| Str^{\pi} \tag{3.13}$$

$$P_i \Rightarrow DS^w(i), i = 1,\ldots L_{head} + L_S \tag{3.14}$$

3.3 水印表示

下面介绍如何利用包模式来表示 0-1 比特序列。标记方与溯源方为了保证包模式的一致性需要对包的时间序列进行同步，即确保公式(3.1)中所有I_i的一致性，于是引入如下假设：

断言 1 标记方发送的数据流的第一个数据包不会丢失，且会作为第一个数据包到达溯源方。

因此，以第一个包的到达时间为相对时间“0”，后续数据包与第一个包的时间差就作为该数据包的相对时间，参数 T 一定时，公式(3.1)中的I_i就独立于自然时间，从而实现标记方与溯源方在数据包时间序列的相对同步，该过程如图 1 所示，标记方与溯源方在不同自然时间观察的一组数据包到达序列，其相对时间实现了同步，这种机制被称为**盲式自同步**(blind self-synchronization)。

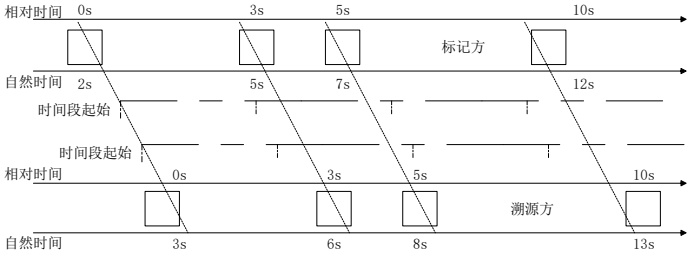


图 1 标记和溯源双方的相对同步

在这个基础上标记方根据不同的比特设置不同的数据包时间序列，从而产生一个包模式P_i。由于一个时间段I_i被用来标记一个比特信息，所以I_i内的包模式P_i可以设置为p_0或者p_1，从标记方的角度来看，可表示为(3.3)的逆：

$$P_i = \begin{cases} p_0 & , DS^w(i)=0 \\ p_1 & , DS^w(i)=1 \end{cases} \tag{3.15}$$

模型中如下定义p_0和p_1：

定义 1(区块)：将时间段I_i分成 R 个子段$I_r^i, r = 1,\ldots,R$，每个子段I_r^i称作一个区块，给定时间t_k^i，显然可以得到该时间所处的区块：

$$r = t_k^i \bmod \frac{I_i}{R} \tag{3.16}$$

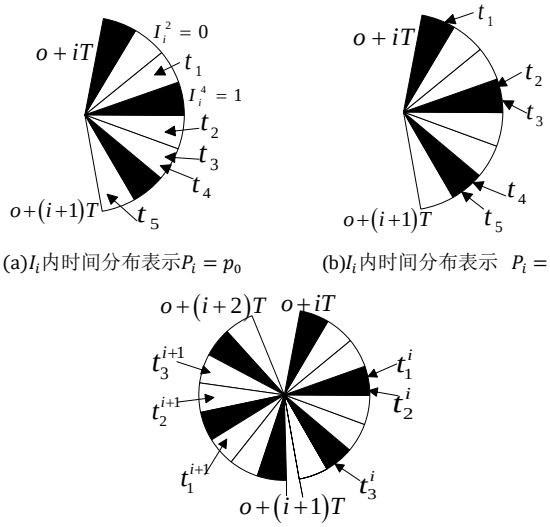
定义 2(区块位图)：将每个区块按照一定规则赋予一个属性值 0 或 1，一个记录每个子段属性值的字典称为区块位图，记为 M，其中所有属性值为 0 的**区块集合**记为M_0，相应的，属性值为 1 的集合记为M_1

定义 3(p_0和p_1)：时间段I_i中，存在数据包模式$P_i = \{t_1^i, t_2^i, \ldots, t_{CS}^i\}$，则有：

$$P_i = \begin{cases} p_0, \sum_{k=1}^{CS} I(t_k^i \in M_0) \geq \sum_{k=1}^{CS} I(t_k \in M_1) \\ p_1, \sum_{k=1}^{CS} I(t_k^i \in M_0) < \sum_{k=1}^{CS} I(t_k \in M_1) \end{cases} \tag{3.17}$$

其中 $I(\cdot)$是二值化符号，如果括号内条件为真，其值为 1，为假则为 0。根据公式(3.17)，可以在标记方设置包模式，在溯源方识别包模式。从定义 3 可以看出，在一个数据包到达的时间序列中，数据包落入哪类区块的数量占优，那么这类区块的属性值就是该包时间序列所表示的比特值。图 2 描绘了某个时间段I_i内两种包模式的相对时间分布，其中黑色的区块属性值为 1，白色的区块属性值为 0，带有区块位图$M = \{1,0,0,1,0,0,1,0\}$，此时I_i内有包序列$P_i = \{t_1^i, t_2^i, \ldots, t_5^i\}$，在图 2(a)中，$P_i$内的所有包都落在了$M_0$中，所以由公式(3.17)可知$I_i$内的包

模式为$P_i = p_0$，类似的，图 2(b)中的包模式为$P_i = p_1$。



(a)I_i内时间分布表示$P_i = p_0$　　(b)I_i内时间分布表示 $P_i = p_1$

(c)相邻的两个时间段I_i和I_{i+1}代表$P_i = p_1$和$P_{i+1} = p_0$

图 2 包模式的表示

至此标记方就可以通过I_i内的包模式P_i标记一位比特信息$bit(I_i)$，继而生成所有时间段内的包模式：

$$Seq. = \left\{ P_i \mid P_i = \left\{ t_1^i, t_2^i, \ldots, t_{CS_i}^i \right\}, i \le L_{head} + L_S \right\} \quad (3.18)$$

对于每个时间段I_i内的数据包个数CS_i，其值可不必都相等但必须是标记方和溯源方约定好的。每一个$t_k^i, k \le CS_i$的取值是从被标记比特对应的区块集合$M_{bit(P_i)}$中等概率的随机选择一个区块，然后放入该区块中。为了提高健壮性，我们只选择每个区块的前半部分，即有：

$$t_k^i = rand(\min(I_r^i), \frac{1}{2}\max(I_r^i)] \quad (3.19)$$

其中，

$$I_r^i = \begin{cases} rand(M_0), if\ P_i \Rightarrow 0 \\ rand(M_1), if\ P_i \Rightarrow 1 \end{cases} \quad (3.20)$$

公式(3.19)中的 *rand(·]*表示从左开右闭区间中随机选择一个数值，公式(3.20)中的 *rand(·)*表示从对应区块集合中随机选择一个区块。如果所有的区块大小相等，溯源方只需知道区块数量R就可以确定每个区块的时间范围。此时t_k^i是相对于I_i的起始时间$o + iT$的，那么在整个数据流的时间序列 *Seq.*中应有：

$$t_k^i = o + iT + rand(\min(I_r^i), \frac{1}{2}\max(I_r^i)] \quad (3.21)$$

引入一个时间刻度参数 α 得到：

$$t_k^i = o + \alpha(iT + rand(\min(I_r^i), \frac{1}{2}\max(I_r^i)]) \quad (3.22)$$

从而得到时间序列：

$$Seq. = \left\{ t_1^1, \ldots, t_{CS_{L_{head}+L_S} \times (L_{head}+L_S)}^{L_{head}+L_S} \right\} \quad (3.23)$$

此时若某个满足条件：

$$N \ge (L_{head} + L_S) \times \sum_i^{L_{head}+L_S} CS_i \quad (3.24)$$

的数据流$PS = \{pt_0, pt_1, \ldots, pt_N\}$已经准备发送，标记方就按照 *Seq.*中的时间序列发送 *PS* 得到$PS^w = \{pt_0^w, pt_1^w, \ldots, pt_N^w\}$：

$$Time(pt_l) = \begin{cases} now\ time\ ,\ if\ l = 0 \\ Seq.(l)\ ,\ if\ l > 0 \end{cases} \quad (3.25)$$

算法 1 标记过程

Input: All parameters referenced in Figure. 3

1: $S^{bit} \leftarrow map(S),\ L_S \leftarrow S^{bit}.length,\ H \leftarrow bin(L_S)$
2: $PTS \leftarrow hash(system.NowTime)$
3: $V^*[] \leftarrow sum(S^{bit}), sum(H)$
4: $S^\pi,\ H^\pi \leftarrow \emptyset$
5: **for all** positions in S^{bit}, H **do**
6: $\quad S^\pi(\pi(j)) \leftarrow S^{bit}(j),\ H^\pi(\pi(h)) \leftarrow H(h)$
7: **end for**
8: $DS^w \leftarrow PTS||H^\pi||S^\pi,\ Seq. \leftarrow \emptyset$
9: **for all** bit in DS^w **do**
10: $\quad P_i \leftarrow \emptyset$
11: $\quad$ **for all** $k \le CS_i$ **do**
12: $\quad\quad t_k \leftarrow t_k.GetRandomTime(I_i\ ,\ M_{bit}),\ P_i \leftarrow P_i \cup t_k$
13: $\quad$ **end for**
14: $\quad Seq. \leftarrow Seq. \cup P_i$
15: **end for**
16: **if** $PS.length < Seq.length$ **then**
17: $\quad$ **return** *FALSE*
18: **end if**
19: **for all** pt_l in *PS* **do**
20: $\quad$ **if** $l = 0$ **then**
21: $\quad\quad pt_0.SendTime \leftarrow system.NowTime$
22: $\quad$ **end if**
23: $\quad pt_l.SendTime \leftarrow Seq.(l)$
24: **end for**
25: Put *PS* on the transmission process
26: **return** *TRUE*

算法 1 描述了整个标记过程，其中第 1~4 行得到了标记过程所有需要的参数，并将字符串通过映射 *map(·)* 转换为比特序列。5~7 行是对比特序列进行位置置换。9~15 行根据每一位比特生成其对应的包模式，16~26 行是按照包模式对应的时间序列发送数据流，最终完成整标记过程。

而溯源方捕获到含有水印的数据流后，根据包模式P_i来溯源一位比特信息$bit(I_i)$，并且利用已知的L_{head}得到头部信息从而溯源出L_{string}。此过程中，公式(3.17)中P_i的数据包数量是溯源方在I_i内观察到的$\widehat{CS_i}$，受丢包、时延等因素的影响，$\widehat{CS_i}$可能与已知的CS_i不同。

算法 2 溯源过程

Input: All parameters referenced in Figure. 3

1: $o \leftarrow pt_0^w$
2: $Seq.,\ PTS||H^\pi,\ S^\pi,\ S^{bit} \leftarrow \emptyset$
3: **for all** $pt_l^w.ReceiveTime$ **do**
4: $\quad Seq. \leftarrow Seq. \cup (pt_l^w.ReceiveTime - o)$
5: **end for**
6: **for all** $q \le L_{head}$ **do**

7: $I_q \leftarrow [o+qT, o+(q+1)T]$

8: $P_q \leftarrow all\ t_k\ in\ I_q$

9: $PST \parallel H^{\pi} \leftarrow PST \parallel H^{\pi} \cup bit(pattern(P_q))$

10: **end for**

11: $L_S \leftarrow base10(H^{\pi}.GetRightPosition())$

12: **for all** $j \le L_S$ **do**

13: $j \leftarrow j + L_{head}$

14: $I_j \leftarrow [o+jT, o+(j+1)T]$

15: $P_j \leftarrow all\ t_k\ in\ I_j$

16: $S^{\pi} \leftarrow S^{\pi} \cup bit(pattern(P_j))$

17: **end for**

18: $S^{bit} \leftarrow S^{\pi}.GetRightPosition()$

19: $S \leftarrow map^{-1}(S^{bit})$

20: **return** S

算法 2 描述了水印溯源的过程，其中 1~5 行溯源方计算观察到的数据流的相对时间序列，6~11 行根据相对时间序列对应的包模式，将水印内容的头部信息还原出来，12~17 行根据头部信息还原出乱序的字符比特序列，18~20 行将乱序的比特流还原为水印内容，完成整个溯源过程。

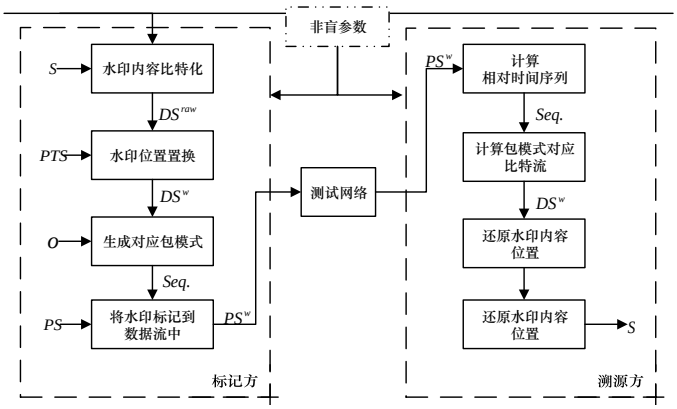


图 3 标记和溯源的不同阶段

最后，图 3 描述了溯源和标记过程的不同阶段，表格 1 中列出了模型所需要的参数，分为由标记方和溯源方事先约定好的非盲参数，和由标记方生成，溯源方通过水印获得的盲参数。

表格 1 模型中的水印参数

非盲参数	
$map(\cdot)$	私有字符串映射函数
L_{head}	头部信息长度
CS_i	单时段标准数据包数量
T	时间段长度
K_t	安全密钥
R	区块数量
M	区块位图
$\pi(\cdot)$	置换函数
α	时间刻度

(a)非盲参数，在编溯源两方都是安全、准确的

盲参数	
S	字符串内容
L_S	字符串长度
o	基准时间
V^*	比特1数量
PTS	伪时间戳

(b)盲参数，在水印传输时可被探测、干扰

4 模型分析

4.1 水印内容还原正确率

由于网络流水印模型中的水印是以比特为单位标记的，因此单个比特的溯源成功率决定整个水印内容溯源成功率。假设溯源方观察到：

$$I_i \Rightarrow P_i = \left\{ t_1^i, t_2^i, \ldots, t_{CS_i}^i \right\} \tag{4.1}$$

其中$\widehat{CS}_i$表示$\hat{P}_i$中数据包的个数。如果数据包在传输过程中只发生延时，而没有PS^w之外的数据包引入，那么当给定一个位于当前时间段内的时刻$\hat{t}_k^i$时，它由两种可能的方式产生：由当前时间段内的数据包延时产生或从上一个时间段的数据包经过时延落到了当前时间段。据此可以计算某个数据包位于正确区块上的概率：

$$\begin{aligned} \mathrm{P}_1 &= p(t_k^i \in M_{bit(P_i)}, \sigma) \\ &= p(t_k^i \in M_{bit(P_i)} \mid \sigma) p(\sigma) \\ &= p(\left\{ t_k^i, t_{k'}^{i-1} \right\} + \sigma \in M_{bit(P_i)}) p(\sigma) \\ &= \sum_m p(\sigma_m) + \sum_{m'} p(\sigma_{m'}) \end{aligned} \tag{4.2}$$

其中，

$$\begin{aligned} & i \ge 1 \\ & m \in \{ m \mid t_k^i + \sigma_m \in \bigcup_{r \ge \lceil t_k^i \rceil}^{R} I_r^i \} \\ & m' \in \{ m' \mid t_{k'}^{i-1} + \sigma_{m'} \in M_{bit(P_i)} \} \end{aligned} \tag{4.3}$$

随机变量σ表示具有概率密度Ω的网络延迟，$r \ge \lceil t_k^i \rceil$表示位于原数据包所在区块之后的区块。则当前$\hat{P}_i$中位于正确区块的数据包数量$\omega$服从二项式分布：

$$\omega \sim B(CS_i, \mathrm{P}_1) \tag{4.4}$$

若要使$\hat{P}_i$能够正确的还原出标记方通过P_i标记的比特，需有:

$$\omega > \left\lfloor \frac{CS_i}{2} \right\rfloor \tag{4.5}$$

利用**正则不完全贝塔函数**[18]的性质有：

$$\mathrm{P}_{P_i = \hat{P}_i} = p(\omega > \left\lfloor \frac{CS_i}{2} \right\rfloor) = I_{AC_p}(\left\lfloor \frac{CS_i}{2} \right\rfloor + 1, CS_i - \left\lfloor \frac{CS_i}{2} \right\rfloor) \tag{4.6}$$

得到成功溯源完整水印内容的概率：

$$\mathrm{P}_{PTS} = \left(\mathrm{P}_{\hat{P}_i = P_i} \right)^{L_{PTS}} \tag{4.7}$$

$$\mathrm{P}_H = \mathrm{P}_{PTS}(\mathrm{P}_{P_i=P_i})^{L_H} \tag{4.8}$$

$$\mathrm{P}_S = \mathrm{P}_H(\mathrm{P}_{P_i=P_i})^{L_S} \tag{4.9}$$

4.2 安全性分析

含有水印的数据流PS^w经过中间网络时，攻击者可能会获得其时间序列。然而，如果缺少了表格 1 中的参数，攻击者便无法窃取和更改水印内容，但可以对监听到的PS^w添加干扰，使溯源方无法得到PS^w中的水印内容。下面证明本文提出的网络流水印模型在一定程度内可以抵抗以下干扰：

(a)丢弃：攻击者从PS^w中随机的去除一些数据包，将剩余的数据流$\widehat{PS^w}$发送给溯源方

此时公式(4.2)、公式(4.4)、公式(4.6)依然成立，所以，只要攻击者不知道表格 1 中的参数，并且没有将某个时间段内I_i所有的数据包去除，溯源方依然能够以概率$P_{\hat{P}_i=P_i}$正确的还原I_i内的比特。若攻击者随机去除长度为 N，含有水印的数据流PS^w中的 ξ 个数据包，则导致$\widehat{PS^w}$中I_i内没有数据包的概率为：

$$\mathrm{P}_{\hat{I}_i=\varnothing} = \frac{\binom{N-CS}{\xi-CS}}{\binom{N}{\xi}} = \frac{\xi!(N-CS)!}{N!(\xi-CS)!}, N \geq \xi \geq CS \tag{4.10}$$

当 ξ 一定时只要增大 CS，公式(4.10)中的概率就会大大减小。

(b)注入：攻击者在PS^w中随机添加一些数据包

如果攻击者没有破坏头部信息，即公式(4.8)中的事件成立，溯源方可以计算出数据流$\widehat{PS^w}$中$\hat{N}$的上限：

$$N \leq (L_{head} + L_S) \times CS \times \tau \tag{4.11}$$

其中 τ 是和映射函数 $map(\cdot)$有关的字符到比特的变化刻度，因此，只要根据公式(4.11)就可以轻易的判断是否有非法的数据包注入到$\widehat{PS^w}$中。如果攻击者在头部信息中注入了数据包，并且没有超出上界：

$$CS_i \leq i \times CS \tag{4.12}$$

那么根据公式(4.2)就变成：

$$\begin{aligned} \mathrm{P}_1 &= p(t_k^i \in M_{bit(P_i)}, \sigma) + p(\overset{+}{t_k^i} \in M_{bit(P_i)}) \\ &= \mathrm{P}_1 + \frac{\left|M_{bit(P_i)}\right|}{|M|} \end{aligned} \tag{4.13}$$

将公式(4.13)代入公式(4.6)得到

$$\begin{aligned} \mathrm{P}_{P_i=P_i} &= p(\omega > \left\lfloor \frac{\Delta}{2} \right\rfloor) = I_{\mathrm{P}_1}(\left\lfloor \frac{\Delta}{2} \right\rfloor + 1, \Delta - \left\lfloor \frac{\Delta}{2} \right\rfloor) \\ \Delta &= CS_i + CS_i^+ \end{aligned} \tag{4.14}$$

其中CS_i^+表示攻击者在时间段I_i内注入的数据包数量，最后根据公式(4.8)我们依然有：

$$\mathrm{P}_H = (\mathrm{P}_{P_i=P_i})^{L_H} \tag{4.15}$$

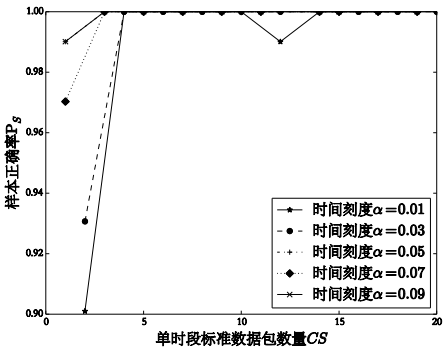
这样，在公式(4.15)所计算的概率的条件下，我们可以通过公式(4.11)来检测到攻击者的注入行为。

(c)重放攻击：攻击者通过监听获得数据流PS_Δ^w的时间序列$Seq._\Delta$，并将另一不同的流PS_∇^w的时间序列$Seq._\nabla$替换为$Seq._\Delta$

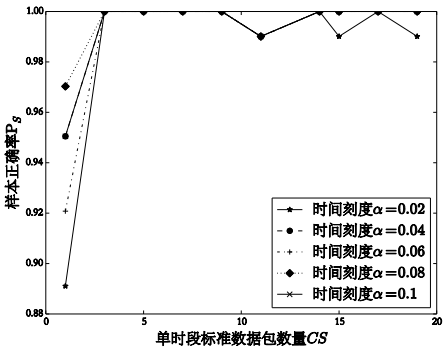
抵抗这种攻击的常用方法是添加时间戳，溯源方根据数据流的时间戳计算与当前时刻的差值，如果小于一定阈值则认为该数据流是重放的数据流。但是，如果标记方将发送的数据流打上有规律的时间戳，那么便会遭到多流攻击。于是溯源方约束伪时间戳 PTS 的内容只能被标记一次，但这样也会带来资源消耗的增加。

(d)多流攻击：攻击者通过观察多条被标记了相同水印内容的数据流来检测流水印的存在，恢复被标记数据流中所的水印内容和相关参数，甚至将水印内容从被标记的数据流中移除。

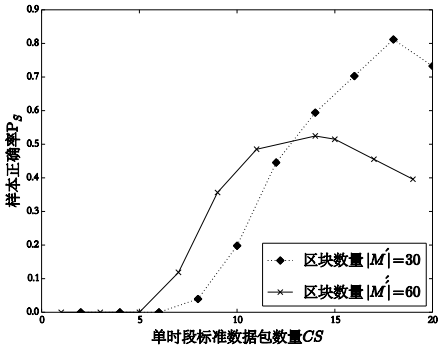
文献[13]和[19]证明了如果针对不同的流标记不同的内容以及改变水印在流中的位置，可以避免遭到多流攻击。显然通过公式(3.4)、公式(3.11)、公式(3.7)可以看出模型完全可以抵御多流攻击。



(a) 实验 *T1* 使用区块位图M'的正确率



(b) 实验 *T2* 使用区块位图M''正确率



(c) 关于参数 α=0.001 的对照实验组

图 4 两组实验在不同参数组合下的溯源正确率

5 实验验证

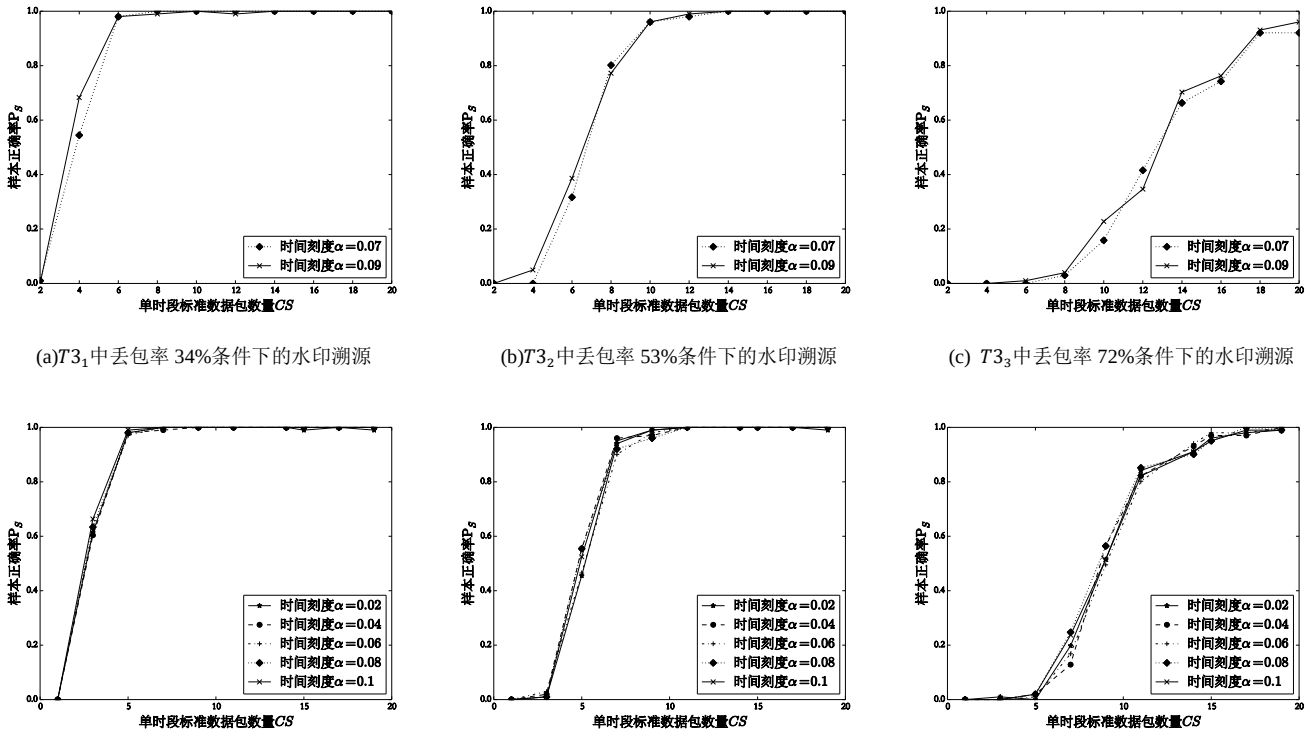
实验环境设置了标记方、溯源方、攻击者来验证网络流水印模型对测试流量进行标记和溯源的效果。实验程序部署在 Linux 系统的节点上，并通过 *netfilter iptables*[20]来调整数据包之间的时间间隔。

5.1 参数选择

首先选取了两种不同的区块位图M'和M''(具有相同时间段长度 T)，其中$|M'|=30$，$|M''|=60$表示区块的数量，进行了两组实验 *T1* 和 *T2*。每组实验使用不同的时间刻度参数 α 和单时段标准数据包数量CS_i，为了实验的方便性，同一数据流内每个时间段内的CS_i是相等的。*T1* 和 *T2* 各测试了 50 个 α 和 *CS* 的参数组合，每个组合通过 3 个标记方节点发出的共计 101 条含有不同水印内容的数据流验证。1 个溯源方节点计算每个参数组合的正确率，即 101 条数据流中完整溯源到水印内容的条数的比例。实验结果如图 4 所示，两组实验的溯源正确率都达到了 89%以上，而且 *CS* 大于 3 的时候，正确率就可以达到 100%。图 4(a)中基于M'的实验结果相对(b)更加稳定，这是因为$|M'|$较小，每个区块的长度较大，减缓了数据包延迟带来的区块偏移。图 4(c)中 α 小于(a)、(b)一个数量级，其结果很差，因为较小的时间刻度这会将传输过程中的扰动放大。

5.2 抗丢包

本实验在 5.1 节实验的基础上，增加了一个攻击者节点，该节点会以一定的比例随机丢弃数据包。测试的两组实验记作 *T3* 和 *T4*，每组实验总共在不同丢包率的环境下测试了 3 次，记为$T3_1$、$T3_2$、$T3_3$和$T4_1$、$T4_2$、$T4_3$，实验结果如图 5 所示。整体上，只要增大 *CS*，丢包后溯源的正确率就会提高，图 5(d)丢包率 25%的情形下，*CS* 为 5 的时候正确率就已经接近了 100%，图 5(c)中，在丢包率高达 72%的情形下，*CS* 大于 18 时正确率依然能达到 90%以上。另外丢包率超过 25%的时候，时间刻度 α 的微小变化带来的影响几乎可以忽略不计，因为图 5(d)(e)(f)中，所有的曲线几乎都重合在一起。而图 5(a)(b)(c)和(d)(e)(f)的对比表明区块数量的调整在丢包的环境下没有起到减缓作用。所以，面对数据包的丢失，使用冗余方式去标记水印内容依然是最为有效的手段。



(a)$T3_1$中丢包率 34%条件下的水印溯源　(b)$T3_2$中丢包率 53%条件下的水印溯源　(c) $T3_3$中丢包率 72%条件下的水印溯源

(d) $T4_1$中丢包率 25%条件下的水印溯源　(e) $T4_2$中丢包率 45%条件下的水印溯源　(f) $T4_3$中丢包率 63%条件下的水印溯源

图 5 不同丢包率、不同参数下的模型溯源正确率实验结果

6 结论

为了在流量测试床中对测试流量进行标记和溯源，本文提出了一种基于时间间隔的网络流水印标记和溯源模型。该模型通过对测试流量标记 0-1 比特序列的水印实现了任意内容的安全传输，同时，理论分析证明了该模型能够抵御多种已知的攻击手段，更重要的是，大量的实验结果证明该模型在高延迟、高丢包率的环境下依然能够维持稳定的溯源正确率。

参考文献

[1] CHUN B, CULLER D, ROSCOE T, et al. Planetlab: an overlay testbed for broad-coverage services [J]. ACM SIGCOMM Computer Communication Review, 2003, 33(3): 3-12.

[2] BERMAN M, CHASE J S, LANDWEBER L, et al. Geni: A federated testbed for innovative network experiments [J]. Computer Networks, 2014, 61: 5-23.

[3] CHANG X. Network simulations with OPNET[C]//Proceedings of the 31st conference on winter simulation: Simulation---a bridge to the future-Volume 1. ACM, 1999: 307-314.

[4] BHATIA S, MOTIWALA M, MUHLBAUER W, et al. Trellis: A platform for building flexible, fast virtual networks on commodity hardware[C]//Proceedings of the 2008 ACM CoNEXT Conference. ACM, 2008: 72.

[5] PAXSON V, FLOYD S. Why we don't know how to simulate the Internet[C]//Proceedings of the 29th conference on winter simulation. IEEE Computer Society, 1997: 1037-1044.

[6] BAJAJ S, BRESLAU L, ESIN D, et al. Improving simulation for network research [J]//Technical Report 99-702b, USC Computer Science Department, 1999.

[7] WANG X, CHEN S, JAJODIA S. Network flow watermarking attack on low-latency anonymous communication systems[C]//Security and Privacy, 2007. SP'07. IEEE Symposium on. IEEE, 2007: 116-130.

[8] WANG X, REEVES D S. Robust correlation of encrypted attack traffic through stepping stones by manipulation of interpacket delays[C]//Proceedings of the 10th ACM conference on Computer and communications security. ACM, 2003: 20-29.

[9] PYUN Y J, PARK Y H, WANG X, et al. Tracing traffic through intermediate hosts that repacketize flows[C]//INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE. IEEE, 2007: 634-642.

[10] HOUMANSADR A, KIYAVASH N, BORISOV N. RAINBOW: A Robust and Invisible Non-Blind Watermark for Network Flows[C]//NDSS. 2009.

[11] SULTANA S, SHEHAB M, BERTINO E. Secure provenance transmission for streaming data [J]. Knowledge and Data Engineering, IEEE Transactions on, 2013, 25(8): 1890-1903.

[12] HOUMANSADR A, BORISOV N. SWIRL: A Scalable Watermark to Detect Correlated Network Flows[C]//NDSS. 2011.

[13] KIYAVASH N, HOUMANSADR A, BORISOV N. Multi-flow Attacks Against Network Flow Watermarking Schemes[C]//USENIX Security Symposium. 2008: 307-320.

[14] ZHANG Y, PAXSON V. Detecting Stepping Stones[C]//USENIX Security Symposium. 2000, 171: 184.

[15] REED M G, SYVERSON P F, GOLDSCHLAG D M. Anonymous connections and onion routing[J]. Selected Areas in Communications, IEEE Journal on, 1998, 16(4): 482-494.

[16] ROOSTA T G. Attacks and defenses of ubiquitous sensor networks [M]. ProQuest, 2008.

[17] LUO X, ZHOU P, ZHANG J, et al. Exposing invisible timing-based traffic watermarks with BACKLIT[C]//Proceedings of the 27th Annual Computer Security Applications Conference. ACM, 2011: 197-206.

[18] PARIS, R. B., "Incomplete beta functions"[M]// in Olver, Frank W. J.; Lozier, Daniel M.; Boisvert, Ronald F.; Clark, Charles W., NIST Handbook of Mathematical Functions, Cambridge University Press, ISBN 978-0521192255, MR 2723248.

[19] HOUMANSADR A, KIYAVASH N, BORISOV N. Multi-flow attack resistant watermarks for network flows[C]//Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on. IEEE, 2009: 1497-1500.

[20] KADLECSIK J, WELTE H, MORRIS J, et al. (2015), The netfilter/iptables Project [EB/OL]. //URL: http://www. netfilter. org.

[21] PENG P, NING P, REEVES D S. On the secrecy of timing-based active watermarking trace-back techniques[C]//Security and Privacy, 2006 IEEE Symposium on. IEEE, 2006: 15 pp.-349.