

# Large-Scale IP Traceback in High-Speed Internet: Practical Techniques and Theoretical Foundation

Jun Li    Minho Sung    Jun (Jim) Xu  
College of Computing  
Georgia Institute of Technology  
{junli,mhsung,jx}@cc.gatech.edu

Li (Erran) Li  
Bell Labs  
Lucent Technologies  
erranli@bell-labs.com

Qi Zhao  
College of Computing  
Georgia Institute of Technology  
qzhao@cc.gatech.edu

**Abstract**—Tracing attack packets to their sources, known as IP traceback, is an important step to counter distributed denial-of-service (DDoS) attacks. In this paper, we propose a novel packet logging based (i.e., hash-based) traceback scheme that requires an order of magnitude smaller processing and storage cost than the hash-based scheme proposed by Snoeren et al. [20], thereby being scalable to much higher link speed (e.g., OC-768). The baseline idea of our approach is to sample a small percentage (e.g., 3.3%) of packets and store their Bloom filter digests. The challenge of this low sampling rate is that much more sophisticated “signal processing” techniques need to be used for traceback, since the “signal strength” becomes much weaker compared to [20]. Our solution is to construct an attack tree by detecting the “statistical correlation” between neighboring routers. However, with naive independent random sampling, our scheme does not perform well due to the low statistical correlation between neighboring routers. We invent a sampling scheme that improves this correlation and the overall efficiency by orders of magnitude. Another major contribution of this work is that we introduce a novel information-theoretic framework to answer important questions on system parameter tuning and the fundamental trade-off between the resource used for traceback and the traceback accuracy. Simulation results based on real-world network topologies (e.g. Skitter) matches very well with results from the information-theoretic analysis. The simulation results also demonstrate that our traceback scheme can achieve high accuracy, and scale very well to a large number of attackers (e.g., 5000+).

**Index Terms**—IP traceback, distributed denial-of-service attacks, system design, information theory

## I. INTRODUCTION

Distributed Denial of Service (DDoS) attacks against high-profile web sites such as Yahoo, CNN, Amazon and E\*Trade in early 2000 [13] demonstrate how damaging DDoS attacks are, and how defenseless the Internet is under such attacks. The services of these web sites were unavailable for hours or even days as a result of the attacks. New instances of DDoS attacks continue to be reported. For example, a recent DDoS attack against root DNS servers brought down eight of them in an effort to

paralyze the majority of Internet service [17]. It is clear that DDoS attacks will not stop or scale down until they are properly addressed.

One possible way to counter DDoS attacks is to trace the source of attacks and punish the perpetrators. However, current Internet design makes such tracing difficult in two aspects. First, there is no field in the IP header that indicates its source except for the IP address, which can be easily spoofed by an attacker. Second, the Internet is stateless in that it keeps no track of the path a packet has traversed. Recently, efforts are made to change one or both aspects to allow for tracing packets to their sources, known as *IP Traceback*. Up to now, two main types of traceback techniques have been proposed in the literature.

- 1) One is to mark each packet with partial path information probabilistically [9], [19], [21], [8], [14]. By receiving a significant number of packets, the victim can construct the attack paths of attackers. This is referred to as Probabilistic Packet Marking (PPM) scheme.
- 2) The other is to store packet digests in the form of Bloom filters [3] at each router [20]. By checking upstream routers iteratively with attack packets, the attack path of a flow can be constructed. This is referred to as hash-based scheme.

However, both traceback schemes suffer from scalability problems. As we will show in the next section, PPM schemes can not scale to large number of attackers as the best scheme proposed can only efficiently trace fewer than 100 attackers using a 17-bit marking field (discussed later); Hash-based scheme is not scalable for high-speed links since recording 100% of packets, even in the Bloom filter digest form, would incur prohibitively high computational and storage overhead. *The objective of our work is to design a traceback scheme that is scalable both in terms of the number of attackers and in terms of high link speed.*

### A. Scalability problems of existing approaches

The advantage of PPM schemes is that they do not incur any storage overhead at the routers and the computation

of marking is usually lightweight. However, PPM-based schemes work well only when the number of attackers is small, due partly to the limited number of bits available for marking in the IP header field. A recent PPM scheme proposed by Goodrich [14] is shown to be the most scalable<sup>1</sup> among the PPM schemes. However, with a marking field of 17 bits, it can only scale up to attack trees containing 100 routers<sup>2</sup>. A large-scale DDoS attack can have thousands of attackers and tens of thousands of routers on the attack paths, making the PPM schemes unsuitable for large-scale traceback.

Hash-based approach, on the other hand, is very effective for large-scale IP traceback, and needs only a single packet<sup>3</sup> to trace one attacker [20]. However, since it computes and stores a Bloom filter digest for every packet, its computational and storage overhead is prohibitive at a router with links of very high speed. For example, assuming a conservative packet size of 1,000 bits, a duplex OC-192 link requires 60 million hash operations to be performed every second, resulting in the use of SRAM (50ns DRAM is too slow for this) and 44 GB of storage space every hour, with the parameters suggested in [20]. It is important to reduce the computational, memory and storage overhead of the hash-based scheme for it to be practical for high-speed Internet.

### B. Our solution for scalable large-scale traceback

In this paper we propose a new traceback scheme that is scalable both to a large number of attackers and to high link speeds. Like [20], our scheme requires Internet routers to record Bloom filter digests of packets going through them. However, unlike [20], which records 100% of packet digests, our scheme only samples a small percentage of them (say 3.3%) and stores the digests of the sampled packets. With such a sampling rate, the storage and computational cost becomes much smaller, allowing the link speed to scale to OC-192, or even higher rates like OC-768. For example, our scheme can scale to OC-768 speed (simplex) using only DRAM, when sampling 3.3% of the traffic<sup>4</sup>.

The trade-off of sampling is that it makes the traceback process much more difficult, especially with a low sam-

pling rate such as 3.3%. In particular, it is no longer possible to trace one attacker with only one packet. This is because, due to sampling, the probability that two neighboring routers on an attacker’s path both sample this packet, is very small. This makes the one-packet traceback operation hard to proceed.

In our scheme, the victim uses a set  $L_v$  of attack packets it has received as “material evidence” to trace and construct the attack tree, consisting of attack paths from attackers to the victim. We will show that the size of  $L_v$  is reasonable even when there are a large number of attackers, although this size is expected to be much larger than used in [20], due to sampling. The tree starts with the victim as the root and the only leaf. It grows when a leaf node  $R_1$  determines that one or more of its upstream neighbors are highly likely to be on an attacker’s path (called “infected” thereafter). Such a likelihood is assessed by performing the following statistical test. Suppose  $R_1$  is a leaf node that is already considered as being infected (called “convicted”).  $R_1$  would like to check whether one of its upstream neighbors  $R_2$  is likely to be on the attack path. We define “what  $R_1$  has seen” as the packets among  $L_v$  that match the Bloom filter digests stored at  $R_1$ . Our statistical test is to check whether “what  $R_1$  has seen” has non-negligible statistical correlation with “what  $R_2$  has seen”, as determined by a threshold decoding rule. If the answer is yes,  $R_2$  will be convicted; Otherwise,  $R_2$  will be exonerated. If  $R_2$  is convicted and it is not yet the last hop to the attacker,  $R_2$  will further test its upstream neighbors recursively using this procedure. Designing the aforementioned threshold decoding rule is nontrivial, and careful game-theoretical study is needed to make sure that the rule is loophole-free to the attackers.

Clearly, the higher the statistical correlation between the attack packets sampled by neighboring infected routers is, the more accurate our traceback scheme is. Other parameters such as sampling rate and the number of attack packets gathered by the victim (i.e.,  $|L_v|$ ) being fixed, it is critical to improve the *correlation factor*, the percentage of the attack packets sampled by  $R_2$  (upstream) matched by the attack packets sampled by  $R_1$  (downstream). A naive sampling scheme is that each router independently samples a certain percentage (say 3.3%) of packets. However, in this case the correlation factor of two routers is just 3.3%. In other words, “what  $R_1$  has seen” only matches 3.3% of “what  $R_2$  has seen”. While consistent sampling techniques such as trajectory sampling [10] has the potential to improve this factor to nearly 100%, it will not work for an adversarial environment, as we will show. We propose a novel technique that improves this correlation factor significantly, using only one bit in the IP header for communications between neighboring routers to coordinate the sampling. This scheme is shown to be robust against attackers’ tampering.

<sup>1</sup>Song *et al.*’s scheme [21] allows traceback to a large number of attackers. However, it requires the knowledge of the router-level Internet topology, which may not be practical. For the traceback to be tamper-resistant, it also requires most of the Internet routers authenticate to the victim, which can be complicated to deploy and administer.

<sup>2</sup>We assume that the “message size” (defined in [14]) is 64 bits for representing the IP address of the current router and the previous router, and the “collision size” (defined in [14]) is no more than 2.

<sup>3</sup>We note that one packet traceback is often unnecessary since the victims typically receive a decent amount of attack traffic, more than enough for traceback.

<sup>4</sup>Each Bloom filter digest uses 12 hash functions. The reason why we use 12 will be clear in Section III.

Using this technique, our scheme requires much smaller number of attack packets for traceback, and achieves better traceback accuracy, than independent sampling.

The proposed scheme is simulated on three sets of real-world Internet topologies with varying operating parameters. Simulation results demonstrate that, even when there are a large number of attackers, our traceback scheme can accurately find most of them using a reasonable number of attack packets. For example, with a sampling probability of only 3.3%, our traceback scheme can identify 90% of infected routes, using only a total of 175,000 attack packets<sup>5</sup> for traceback (resulting in a query size of 4.9 MB), even when there are 1,000 attackers. Sampling greatly reduce the computational and storage overhead for packet logging. With a sampling rate of 3.3% (it can be smaller), our storage overhead is only  $0.4/\ln 2$  bits per packet<sup>6</sup>, a duplex OC-192 link will require the computation of at most 8 million hash functions every second, and the storage of 5.2 GB for one hour’s traffic. This is an order of magnitude more affordable than [20].

### C. Theoretical foundation of our traceback scheme

The design of the scheme leads to a very interesting optimization problem. We assume that the average number of bits devoted for each packet is a fixed constant  $s$ , due to computational and storage constraints of a router. Then the size of each Bloom filter digest our scheme computes for each sampled packet is inversely proportional to the percentage of packets that is sampled. For example, if the resource constraint is 0.4 bits of computation per packet, one possible combination is that the router samples 5% of the packets and the size of each digest is 8 bits ( $5\% * 8 = 0.4$ ). However, with the same resource constraint, an alternative combination is to sample 2.5% of the packets, but each digest is 16 bits. Which one is better? Intuitively, higher sampling rate increases the aforementioned correlation between two routers, making traceback easier. However, the size of a digest would have to be proportionally smaller, which results in a higher false positive rate in Bloom filter. This adds noise to the aforementioned statistical inferring process and reduces the accuracy. Clearly there is an inherent trade-off between these two parameters, but where is the “sweet spot” (i.e., optimal parameter setting)? We show that the answer to this question lies in the information theory by viewing the traceback system as a communication channel. The optimal parameter setting is exactly the one that maximizes the Shannon capacity of this channel. Our simulation results show that the information-theoretic framework indeed guides us to find the optimal parameter setting!

<sup>5</sup>Only the first 28 invariant bytes of a packet will be used for traceback as in [20].

<sup>6</sup>The term  $\ln 2$  is due to the Bloom filter space-efficiency trade-off and will be explained in Section II-B.2

Our information-theoretic framework also allows us to answer another important question concerning the trade-off between the amount of evidence the victim has gathered (the number of attack packets) and the traceback accuracy. In particular, information theory allows us to derive a lower bound on the number of packets the victim must obtain to achieve a certain level of traceback accuracy. A bonus from studying these lower bounds is that it sheds light on how this number scales to larger number of attackers.

The rest of the paper is organized as follows. In Section II, we articulate the challenge raised by sampling, and describe the components of our scheme in detail. In Section III, the proposed scheme is analyzed using a novel information-theoretic framework. The performance is evaluated in Section IV through simulation studies. Section V surveys the related work and Section VI concludes the paper.

## II. DETAILED DESIGN

Our scheme consists of two algorithms. One is a sampling algorithm that is running at the Internet routers to sample and record the Bloom filter digests of the packets going through them. The other is a traceback algorithm that is initiated by the victim to trace the attackers using the digests stored at these routers, upon the detection of a DDoS attack. In the following, we first present in Section II-A the technical challenges of traceback due to sampling (with small probability) to motivate our design choices. In Sections II-B and II-C, we describe the sampling algorithm and the traceback algorithm in detail.

### A. Design challenges

Our proposed scheme significantly reduces the processing and storage requirement by sampling. However, sampling makes the traceback more difficult. In particular, it is now almost impossible to trace one attacker with only one packet as in [20]. This is because, with a low sampling percentage, the first router on the attack path that will sample a particular attack packet is on the average many hops away. Intuitively, with a sampling rate of  $p$ , the victim needs to receive at least  $O(\frac{1}{p})$  packets to be able to trace one attacker, since each router on the path needs to store at least one attack packet. However, it turns out that to design a sampling algorithm that allows for accurate traceback of one attacker with this minimum number of attack packets (i.e.,  $O(\frac{1}{p})$ ) is nontrivial.

A naive sampling scheme is that each router independently samples packets with the probability  $p$ . However, this approach does not work well since it would require a

minimum of  $O(\frac{1}{p^2})$  attack packets<sup>7</sup> to trace one attacker. Recall from Section I-B that if a convicted router  $R_1$  wants to check whether one of its upstream neighbors  $R_2$  is infected, the scheme checks whether the set of packets “ $R_1$  has seen” has non-negligible correlation with the set of packets “ $R_2$  has seen”. It can be shown that it takes at least  $O(\frac{1}{p^2})$  packets for these two sets to have an overlap of one or more packets. The key problem of this naive scheme is that the correlation factor between neighboring routers is only  $p$ , i.e., “what  $R_1$  has sampled” only matches  $p$  (percentage) of “what  $R_2$  has sampled”. We propose a novel sampling scheme that improves this correlation factor to over 50% with the same sampling rate  $p$  at every router, therefore reaching the  $O(\frac{1}{p})$  asymptotic lower bound. We will discuss this scheme in detail in Section II-B.1.

One may say that there is a scheme that achieves the correlation factor of 100%, by asking all routers on the same path to sample the same set of packets (known as *trajectory sampling* [10]). However, techniques to achieve such consistent sampling will not work in this adversarial environment since an attacker can easily generate packets that evade being sampled by the scheme in [10]. We explored along this direction and found that it is extremely challenging to design noncryptographic<sup>8</sup> techniques to achieve consistent sampling in this adversarial environment. Our scheme, on the other hand, is robust against the tampering by the attackers, without resorting to cryptographic techniques (shown in Section II-C).

Due to sampling, some routers that are on the attack path may not be detected. We call these routers *false negatives*. The *false negative ratio* (FNR) of an attack tree constructed by a traceback scheme is defined as the ratio of the number of false negatives to the number of actual infected routers during the attack<sup>9</sup>. Because the use of Bloom filters to store packet digests (details will be discussed in Section II-B.2), a traceback system may identify routers that were not actually on attack paths. We call these routers *false positives*. The *false positive ratio* (FPR) of an attack tree constructed by a traceback scheme is defined as the ratio of the number of false positives to the total number of routers in the attack tree. It is ideal for the traceback scheme to be able to trace most of the attackers (i.e., low FNR), using a moderate number of attack packets. It is in general not necessary for FNR to be zero (i.e., find all attackers) since identifying and removing most of the attackers are effective enough for restoring

<sup>7</sup>Note that  $O(\frac{1}{p^2})$  can be orders of magnitude larger than  $O(\frac{1}{p})$  when  $p$  is small.

<sup>8</sup>This is possible with cryptographic techniques. However, it may involve key distribution and management on hundreds of thousands of Internet routers.

<sup>9</sup>Recall that a router on the attack path of an attacker is called “infected”.

the services being attacked. In addition, incomplete or approximate attack path information is valuable because the efficacy of complementary measures such as packet filtering improve as they are applied further from the victim and closer to the attack source [19].

## B. Sampling

### 1) One-bit Random Marking and Sampling (ORMS):

Recall that independent random sampling method does not work well since the correlation factor between the neighboring routers is only  $p$  (typically less than 5%), the sampling rate. In this section, we present our ORMS scheme that increases this factor by an order of magnitude. The key idea of our scheme is that, besides sampling the packets, a router also probabilistically marks the sampled packets so that the next router on the path, seeing the mark, can coordinate its sampling with the previous router to improve the correlation factor. We will show that a marking field of one bit is enough<sup>10</sup> for the correlation factor to be over 50%. This bit can be easily fit into many possible locations in the IP header (e.g., IP fragmentation fields).

Our ORMS scheme is presented in Figure 1. If an arriving packet has the bit marked, the bit will be *unmarked* and the packet will be stored in Bloom filter digest form. However, if the percentage of packets (denoted as  $r$ ) that are marked among the arriving packets is over  $\frac{p}{2}$ , it must have been tampered by an attacker (explained next). Our scheme will only sample and store the marked packets with probability  $\frac{p}{2r}$ . This is the meaning of “subject to a cap of  $\frac{p}{2}$ ” in line 4 of Figure 1. If an arriving packet is not marked, it will be stored and marked with probability  $\frac{p}{2-p}$ . A router will also measure the percentage of packets coming from itself that is marked. If this percentage is larger or smaller than  $\frac{p}{2}$ , the router will unmark or mark some packets to compensate for that (lines 9 & 10 in Figure 1). This step is important for an invariant of our sampling scheme (shown next) to hold even for the router on the first hop (i.e., jump-start to “stationarity”).

We will show that the following two invariants hold. The first invariant is that exactly  $\frac{p}{2}$  of the packets from a router will be marked. Note that a router on the first hop from the attacker will mark  $\frac{p}{2}$  of the packets (lines 9 & 10 in Figure 1). This argument certainly works for every router, but we would like to show that once the system is “jump-started” to “stationarity”, these two lines will never be executed at later routers. To see this, note that at later routers,  $(1 - \frac{p}{2})$  of the arriving packets are not marked, and among those  $\frac{p}{2-p}(1 - \frac{p}{2}) = \frac{p}{2}$  will be marked. Therefore, once the system is jump-started to stationarity (with  $\frac{p}{2}$  marked), it remains stationary! The second invariant

<sup>10</sup>Using more bits for marking will further improve the correlation factor, but the resulting scheme is more complicated.

**Sampling procedure at router  $R$  (given sampling rate  $p$ ):**

1. **for** each packet  $w$
  2.   **if** ( $w.\text{mark} = 1$ ) **then**
  3.     write 0 into  $w.\text{mark}$ ;
  4.     store the digest of  $w$ , subject to a cap of  $\frac{p}{2}$ ;
  5.   **else**
  6.     with probability  $\frac{p}{2-p}$
  7.     store the digest of  $w$ ;
  8.     write 1 into  $w.\text{mark}$ ;
  9.   **if** (marking percentage is not  $\frac{p}{2}$ ) **then**
  10.    tune it to  $\frac{p}{2}$ ;
- /\* make the process “stationary” \*/

Fig. 1. One-bit random marking and sampling (ORMS) scheme

is that each router, except for the first hop (which may sample less than  $p$ ), will sample exactly  $p$  of the packets. This is because a router will sample all the packets marked by the upstream neighbors ( $\frac{p}{2}$ ), and sample another  $\frac{p}{2}$  of packets that are marked by itself. Finally, it is not hard to verify that, no matter how an attacker manipulates the marking field, the first router on the attacker’s path will sample at least  $\frac{p}{2}$  of the packets coming from the attacker.

Now we quantitatively analyze the benefit of our one-bit marking technique. We claim that the expected correlation factor between two neighboring routers  $R_1$  (downstream) and  $R_2$  (upstream) is  $\frac{1}{2-p}$ , when  $R_2$  is not on the first hop from the attacker. This is because  $R_1$  has sampled all  $\frac{p}{2}$  percentage of packets  $R_2$  has marked, and among another  $\frac{p}{2}$  percentage of packets that  $R_2$  has sampled but unmarked,  $R_1$  samples  $\frac{p}{2-p}$  of them. The total is  $\frac{p}{2}(1 + \frac{p}{2-p})$ , which is  $\frac{p}{2-p}$ . The correlation factor is  $\frac{p}{2-p}$  (sampled by both) divided by  $p$  (sampled by  $R_2$ ), which is  $\frac{1}{2-p}$ . Note that  $\frac{1}{2-p}$  is larger than 50% because  $0 < p < 1$ . This represents an order of magnitude improvement compared to the independent random sampling, when  $p$  is small (say  $< 5\%$ ).

Finally, we would like to show that the  $\frac{1}{2-p}$  correlation factor of our scheme is resistant to tampering by attackers. In other words, an attacker cannot manipulate this factor by marking or unmarking the packets they send. This is because our ORMS scheme is oblivious: the first router that receives the marked packets from an attacker will unmark them and the output packets from the router have exactly  $\frac{p}{2}$  of them marked (i.e., jump-start to stationarity). Later on, as discussed before, the correlation between neighboring routers will always be  $\frac{1}{2-p}$ .

2) *Packet digesting* : Like in [20], we use a space-efficient data structure known as a Bloom filter [3] to record packet digests. A Bloom filter representing a set  $S = \{x_1, x_2, \dots, x_n\}$  of size  $n$  is described by an array  $A$  of  $m$  bits, initialized to 0. A Bloom filter uses  $k$  independent hash functions  $h_1, h_2, \dots, h_k$  with range  $\{1, \dots, m\}$ . In the *insertion phase*, given an element  $x$  to be inserted into a set  $S$ , the bits  $A[h_i(x)]$ ,  $i = 1, 2, \dots, k$ ,

are set to 1. In the *query phase*, to check if an element  $y$  is in  $S$ , we check the value of the bits  $A[h_i(y)]$ ,  $i = 1, 2, \dots, k$ . The answer to the query is *yes* if *all* these bits are 1, and *no* otherwise.

A Bloom filter guarantees not to have any false negative, i.e., returning “no” even though the set actually contains the element. However, it may contain false positives, i.e., returning “yes” while the element is not in the set. The capacity factor  $c$  of a Bloom filter is defined as the ratio of  $m$  to  $n$ . In this paper, we assume the Bloom filter at each router is paged to disk when  $c$  decreases to  $k/\ln 2$ . Then according to [3], the false positive rate of the Bloom filter is no more than  $1/2^k$ . In Sections III and IV, the false positive rate of the Bloom filter is always assumed to be  $1/2^k$  for analysis and performance evaluation purposes.

Note that same as in [20], we use the first 28 invariant bytes of an IP packet as the hash input. These 28 bytes include the invariant portion of the IP header and the first 8 bytes of the payload. In the rest of the paper, when we refer to a packet, we always refer to its first 28 invariant bytes.

### C. Traceback processing

When the victim detects a DDoS attack (through perhaps an IDS system), it will trigger a traceback processing procedure. The victim will first collect a decent number of attack packets, which is not difficult during a DDoS attack. Then it will use all or a part of these packets to track down the attackers. We denote the set of packets that are used for traceback as  $L_v$ . Note that, The size of  $L_v$  is 4.9MB in the scenario discussed in Section I. The size of  $L_v$  is typically between 1MB and 10MB depending on the number of attackers and the traceback accuracy desired.

The traceback procedure starts with the victim checking all its immediate neighbors. For any router  $S$  which is one hop away from the victim, the victim will first query the corresponding (right date and time) Bloom filter at  $S$  with the whole set  $L_v$ . The router  $S$  is added to the attack tree if at least one match is found. If  $S$  is convicted, the set of packets in  $L_v$  that matches the Bloom filter of  $S$  will be assembled into a set  $L_S$ . Each upstream neighbor  $R$  of  $S$  will then be queried by  $L_S$  (not  $L_v$ !). Again, if at least one match is found and  $R$  has not yet been convicted by another router<sup>11</sup>,  $S$  convicts  $R$  and sends  $L_v$  to  $R$ ; Otherwise, nothing needs to be done to  $R$  by  $S$  (other routers may still convict  $R$ ). If  $R$  is convicted,  $R$  will assemble  $L_R$ , which is the set of packets in  $L_v$  that matches the Bloom filter at  $R$ . The set  $L_R$  will then be used by  $R$  to query its upstream routers. This process is repeated recursively upstream until it can not proceed.

<sup>11</sup>This can happen since the Internet is not a tree.

We now discuss the subtleties involved in our traceback processing. In the above algorithm, a router is convicted if the Bloom filter returns “yes” for at least one packet. It is important to use one as the detection threshold. Otherwise, an attacker can send identical packets to avoid detection. This loophole exists because the Bloom filter we use does not count the number of occurrences of a packet<sup>12</sup>. This loophole is closed under our “one packet decoding rule”, since it can be easily verified that an attacker has no incentive to send identical packets anymore from a game-theoretical point of view, since this will only increase its probability of being detected.

Note that our scheme uses  $L_R$  to match the Bloom filter at the upstream neighbors of  $R$  once  $R$  is convicted. A careful reader may wonder why we do not simply use  $L_v$  to query each router. Recall that, Bloom filter can have a false positive probability of  $2^{-k}$  where  $k$  is the number of hash functions used. We will show that a typical  $k$  value is 12. When  $k = 12$  (with a false positive probability  $2^{-12}$ ) and  $|L_v| \gg 5,000$ , more than one false positive will occur with high probability. This will result in almost all Internet routers being convicted. Since  $|L_R|$  is much smaller than  $|L_v|$ , the false positive caused by  $L_R$  is also much smaller.

Now because  $L_R$  is used, the false positive occurs only when a packet falsely matches both Bloom filters, which happens with probability  $2^{-2k}$ . When  $k = 12$ , it takes 16 million packets to result in one false positive on the average. We will typically use much fewer than 1 million for traceback resulting in very low false positive.

### III. AN INFORMATION-THEORETIC FRAMEWORK

In this section we present our information-theoretic framework that serves as the theoretical foundation of our traceback scheme. We first present the problems that are answered by this framework in Section III-A. After briefly introducing the relevant information theory concepts and theorems in Section III-B, we show how they are applied to our context in Section III-C.

#### A. Why do we need a theoretical foundation?

Our information-theoretic framework answers the following two questions concerning parameter tuning and the minimum number of attack packets needed for accurate traceback, respectively.

<sup>12</sup>One can also use counting Bloom filter [11] or Spectrum Bloom filter [6] to record the number of occurrences of a packet. Detection rules based on multiple packets can be designed accordingly. However, these schemes are much more complicated. Also the game-theoretical analysis associated with using the higher threshold is extremely complex.

1) *Parameter tuning*: We have discussed in Section I that given a resource constraint, the number of hash functions in each Bloom filter is inversely proportional to the sampling probability. Clearly there is an optimal trade-off between these two parameters. Information theory will help us find the “sweet spot”.

2) *Minimum number of attack packets needed*: The information-theoretic framework also allows us to answer the following question: “What is the minimum number of attack packets that the victim has to gather in order to achieve a traceback error rate of no more than  $\epsilon$ ?” This information is important because it exhibits the fundamental trade-off between the number of attack packets the victim needs to use for traceback, and the accuracy to be achieved. Our solution to this question also answers a related question: “How does this number (of attack packets) scale with respect to certain system parameters such as the number of attackers?” For example, if the number of attackers grows from 1,000 to 2,000, how many more attack packets does the victim have to use to achieve the same accuracy?

#### B. Information theory background

In this section, we summarize the information theory concepts and theorems that will be used in our later exploration. We first review the concepts of entropy, conditional entropy, and mutual information, which will be used to answer the question raised in Section III-A.1. Then we introduce Fano’s inequality [7], which will be used to answer the question raised in Section III-A.2.

##### 1) Entropy, Conditional Entropy, Mutual Information:

*Definition III.1*: The entropy of a discrete random variable  $X$  is defined as

$$H(X) \stackrel{def}{=} - \sum_{x \in \mathcal{X}} \Pr[X = x] \log_2 \Pr[X = x] \quad (1)$$

where  $\mathcal{X}$  is the set of values that  $X$  can take. The entropy of a random variable  $X$  measures the uncertainty of  $X$ , in the unit of bits.

*Definition III.2*: The conditional entropy of a random variable  $X$  conditioned on another random variable  $Y$  is defined as

$$H(X|Y) \stackrel{def}{=} - \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} (\Pr[X = x, Y = y] \cdot \log_2 \Pr[X = x|Y = y]) \quad (2)$$

where  $\mathcal{Y}$  is the set of values that  $Y$  can take. The concept of conditional entropy arises when we are interested in estimating the value of  $X$ , which can not be observed directly, using the observation of a related random variable  $Y$ . The conditional entropy  $H(X|Y)$  measures how much uncertainty remains for  $X$  given our observation of  $Y$ .

*Definition III.3:* The mutual information  $I(X; Y)$  is defined as  $H(X) - H(X|Y)$ .

Note that before the observation of  $Y$ , the uncertainty of  $X$  is  $H(X)$ . After the observation, this uncertainty goes down to  $H(X|Y)$ . Therefore, the mutual information measures the amount of information we learn about  $X$  from observing  $Y$ .

2) *Fano's inequality:* In our analysis, we would like to estimate the value of  $X$  based on the observation of  $Y$ . Recall that the conditional entropy  $H(X|Y)$  measures how much uncertainty remains for  $X$  given our observation of  $Y$ . Intuitively, the smaller this conditional entropy value is, the more accurate the estimation that can be made is. This intuition is captured by Fano's inequality [7].

Suppose, given an observation of  $Y$ , our estimation of  $X$  is  $\hat{X}$ . We denote  $p_e$  as the probability that this estimation is incorrect, i.e.,  $p_e = \Pr[\hat{X} \neq X]$ . Fano's inequality [7] states the following.

$$H(p_e) + p_e \log_2(|\mathcal{X}| - 1) \geq H(X|Y) \quad (3)$$

Here,  $H(p_e)$  is "overloaded" to stand for the entropy of the indicator random variable  $1_{\{\hat{X} \neq X\}}$ . By (1),  $H(p_e) = -p_e \log_2 p_e - (1 - p_e) \log_2 (1 - p_e)$ . In (3),  $|\mathcal{X}|$  is the number of different values that  $X$  can take. If we are estimating a random variable that will only take 2 possible values (i.e.,  $|\mathcal{X}| = 2$ ), Fano's inequality becomes the following simplified form:

$$H(p_e) \geq H(X|Y) \quad (4)$$

Note that, without loss of generality, we can assume that  $p_e$  is no more than 0.5 (if a binary estimation procedure  $A$  produces wrong result more than half of the time we can simply use  $\bar{A}$ ). So Fano's inequality and the fact that  $H$  is strictly increasing from 0 to 0.5, implies that if we would like the estimation of  $X$  (binary-valued) to have an estimation error no more than  $p_e$ , the conditional entropy  $H(X|Y)$  has to be no more than  $H(p_e)$ .

### C. Applications to our problems

1) *Tuning the parameter  $k$ :* As described in Section II-C, when a (convicted) router  $R_1$  would like to check whether one of its upstream neighbors  $R_2$  is infected, it queries the Bloom filter at  $R_2$  with  $L_{R_1}$ . Here  $L_{R_1}$  is the set of packets that match the Bloom filter at  $R_1$  among the set of packets used for traceback (i.e.,  $L_v$ ). As we discussed before, our resource constraint is  $kp \leq s$ . Here  $s$  is the number of bits of computation devoted to each packet on the average,  $k$  is the number of hash functions in each Bloom filter, and  $p$  is the sampling probability. Clearly, the best performance happens on the curve  $kp = s$ . Since  $s$  is treated as a constant, only one parameter  $k$  needs to be tuned ( $p = s/k$ ). It remains to be found out which  $k$  value

will allow us to determine with best accuracy whether  $R_2$  has been infected.

This problem can be viewed as a channel capacity maximization problem. Informally, we can draw an analogy to Shannon's channel capacity formula  $C = W \log_2(1 + SNR)$ . One can view both  $W$  and  $SNR$  (Signal Noise Ratio) as a function of  $k$ . Decreasing the parameter  $k$  (i.e., increasing the sampling rate  $p$ ) is analogous to increasing  $W$ , but it decreases  $SNR$  since there are more false positives in the Bloom filter. We are tuning  $k$  to strike the best trade-off between  $W$  and  $SNR$ . Note however that this analogy works only at the conceptual level, and the capacity of our "traceback channel" does not follow the Shannon capacity formula literally. Now recall that Shannon's formula is obtained when the mutual information is maximized using techniques from the *calculus of variations*. Therefore, to maximize the capacity of our channel, we only need to maximize the mutual information between what is observed and what needs to be estimated.

In the following, we calculate this mutual information as a function of  $k$ , and compute  $k$  that maximizes this function. We first define some notations.

- $N_p$ : number of attack packets used by the victim for traceback.
- $d_1$ : the percentage of the attack packets that travel through  $R_1$ .
- $d_2$ : the percentage of the attack packets that travel through  $R_2$ .

In the following, we introduce step by step the random variables involved in the mutual information formula. By convention,  $Binom(\mathcal{N}, \mathcal{P})$  represents the binomial distribution with constant parameters, where  $\mathcal{N}$  is the number of trials and  $\mathcal{P}$  is the "success" probability. In some places below, we abuse the  $Binom$  notation slightly to put a random variable in the place of  $\mathcal{N}$ , which will be made mathematically rigorous next. Let  $X$  be a random variable. The rigorous mathematical definition for a random variable  $\mathcal{Y}$  to have the distribution  $Binom(X, \mathcal{P})$  is that, the conditional distribution of  $\mathcal{Y}$  given that  $X = x$  is  $Binom(x, \mathcal{P})$ , and this holds for all values of  $x$  that  $X$  will take. This abuse is not counterintuitive, and makes our reasoning much more succinct.

- Let  $X_{t_1}$  be the number of attack packets sampled by  $R_1$ . It has the probability distribution  $Binom(N_p d_1, p)$ .
- Let  $X_{f_1}$  be the number of false positives when all the attack packets are queried against the Bloom filter at  $R_1$ . Its probability distribution is  $Binom(N_p - X_{t_1}, f)$ . Here  $f = 2^{-k}$  is the false positive rate of the Bloom filter.
- Let  $X_{t_2}$  be the number of attack packets sampled by  $R_2$ . Its probability distribution is  $Binom(N_p d_2, p)$ .
- Let  $Y_t$  be the number of true positives (real match instead of Bloom filter false positives) when the Bloom

filter at  $R_2$  is queried using  $L_{R_1}$ . Its probability distribution is  $\text{Binom}(X_{t_2}, \frac{1}{2-p})$ . The parameter  $\frac{1}{2-p}$  comes from the fact that the correlation factor between neighboring routers is  $\frac{1}{2-p}$  in our ORMS scheme.

- Let  $Y_f$  be the number of false positives when the Bloom filter at  $R_2$  is queried with  $L_{R_1}$ . Its probability distribution is  $\text{Binom}(X_{t_1} + X_{f_1} - Y_t, f)$ .

During the traceback process, we are able to observe the values of the following two random variables:

- $X_{t_1} + X_{f_1}$ : the total number of packets in the packet set  $L_{R_1}$ .
- $Y_t + Y_f$ : the number of positives when the Bloom filter at  $R_2$  is queried with  $L_{R_1}$ .

We are interested in estimating the value of the following random variable  $Z$ , which indicates whether  $R_2$  has stored at least one attack packet in the set of the attack packets used by the victim for traceback.

$$Z = \begin{cases} 1 & \text{if } X_{t_2} > 0 \\ 0 & \text{otherwise} \end{cases}$$

By the information theory, our knowledge about  $Z$  is maximized when the mutual information  $I(Z; \langle X_{t_1} + X_{f_1}, Y_t + Y_f \rangle)$  is maximized. In other words, we would like to compute

$$k^* = \underset{k}{\operatorname{argmax}} I(Z; \langle X_{t_1} + X_{f_1}, Y_t + Y_f \rangle) \quad (5)$$

subject to the constraint  $kp = s$  as discussed before. Note that  $H(Z)$  is a constant given any a priori distribution of  $Z$ . Since  $I(Z; \langle X_{t_1} + X_{f_1}, Y_t + Y_f \rangle) = H(Z) - H(Z|X_{t_1} + X_{f_1}, Y_t + Y_f)$ , (5) is equivalent to minimizing the conditional entropy  $H(Z|X_{t_1} + X_{f_1}, Y_t + Y_f)$ .

The actual formula of  $H(Z|X_{t_1} + X_{f_1}, Y_t + Y_f)$  in terms of system parameters  $N_p$ ,  $d_1$ ,  $d_2$ , and  $k$  is very involved. The details on how to calculate the conditional entropy can be found in Appendix A. We have written a program to compute  $H(Z|X_{t_1} + X_{f_1}, Y_t + Y_f)$  given a set of parameters. Its results are used to plot the figures related to  $H(Z|X_{t_1} + X_{f_1}, Y_t + Y_f)$  in the rest of the paper.

In general, the value of  $H(Z|X_{t_1} + X_{f_1}, Y_t + Y_f)$  not only depends on the parameter  $k$  we would like to tune, but also depends on other parameters such as  $d_1$  and  $d_2$ . We can view the value of  $d_2$  (say  $d_2 = d$ ) as a *targeted level of concentration*. In other words, when  $k = k^*$ , our system is most accurate, among all the  $k$  values, in estimating the value of  $Z$  given that all potential  $R_2$ 's have the concentration  $d$ . In maximizing  $H(Z|X_{t_1} + X_{f_1}, Y_t + Y_f)$ , we assume  $d_1 = d_2$ . This is because, given a typical router-level Internet topology, when we trace routers several hops away from the victim, with good probability  $R_2$  is the only upstream neighbor of  $R_1$  that is infected (i.e.,

no more ‘‘branching’’ upstream). So  $d_1 = d_2 = d$  captures the ‘‘common case’’ we would like to optimize, and here  $d$  is the level of concentration at potential  $R_2$ 's we would like to target. One may wonder if we target a certain concentration, but a different concentration happens during an attack, our  $k^*$  may not be optimal. However, our computation results show that if we target a low concentration such as  $\frac{1}{5000}$ , which approximately corresponds to 5,000 attackers attacking with the same intensity, our  $k^*$  is optimal or close to optimal for other higher concentrations as well. In the other words, the optimality of  $k$  is not sensitive to the concentration values we are targeting.

We illustrates these results in Figure 2. Each curve in Figure 2(c) shows how the value of  $H(Z|X_{t_1} + X_{f_1}, Y_t + Y_f)$  varies with different  $k$  values, given a certain  $N_p$  value (number of attack packets used for traceback). The three curves in this figure corresponds to  $N_p = 250, 000, 375,000, 500,000$  respectively. Here the resource constraint is  $s = 0.4$ . The targeted concentration  $d$  is  $\frac{1}{5000}$ . We can clearly see that the optimal  $k$  value is not sensitive to the parameter  $N_p$ . Given  $d = \frac{1}{5000}$ , Figure 2(c) shows that  $k = 12$  or  $13$  results in the lowest value for  $H(Z|X_{t_1} + X_{f_1}, Y_t + Y_f)$ .

Figures 2(a) and 2(b) show how the value of  $H(Z|X_{t_1} + X_{f_1}, Y_t + Y_f)$  varies with different  $k$  values, when  $d$  is set to  $\frac{1}{1000}$  and  $\frac{1}{2000}$ , respectively. From these two figures, we see that  $k = 12$  is very close to optimal for higher concentrations  $\frac{1}{1000}$  and  $\frac{1}{2000}$ . This demonstrates that the optimal value of  $k$  is not very sensitive to the value of  $d$ . Therefore, in Section IV, our scheme will adopt  $k = 12$  when its resource constraint is  $s = 0.4$ . We will show that  $k = 12$  indeed allows our scheme to achieve the optimal performance. In other words, the information theory indeed prescribes the optimal parameter setting for our scheme!

2) *Application of Fano's inequality:* In this section, we will show how Fano's inequality can be used to compute the minimum number of attack packets needed for achieving a certain traceback accuracy and how this number scales to larger number of attackers. According to Fano's inequality for the estimation of a binary-valued random variable (formula (4)), we have

$$H(p_e) \geq H(Z|X_{t_1} + X_{f_1}, Y_t + Y_f). \quad (6)$$

where  $p_e = \Pr[\hat{Z} \neq Z]$  is the probability that our estimation  $\hat{Z}$  is different from the actual value of  $Z$ . Note that here  $p_e$  includes both the false positives (i.e., routers wrongly convicted) and the false negatives. Therefore, given a desired traceback error rate  $\epsilon$ , the number of attack packets has to be larger than  $N_{min}$ , where  $N_{min}$  is the minimum  $N_p$  that makes  $H(Z|X_{t_1} + X_{f_1}, Y_t + Y_f)$  no more than  $H(\epsilon)$ .

Figure 3 shows the fundamental trade-off between the traceback error  $p_e$  and  $N_{min}$ . In this figure,  $s$  is set to

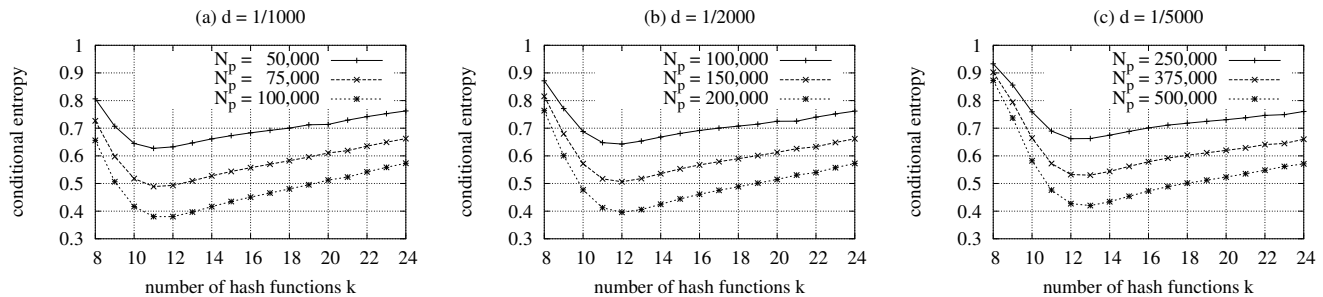


Fig. 2. Conditional entropy with respect to the number of hash functions used in a Bloom filter for  $s = 0.4$  with different concentrations.

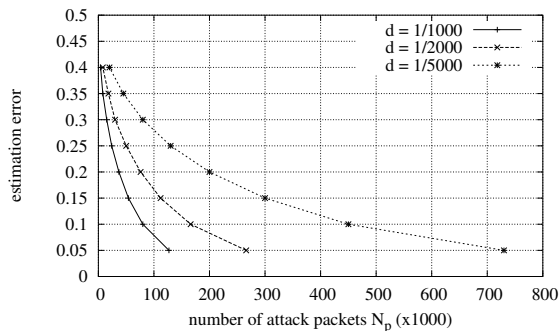


Fig. 3. The trade-off between the estimation error  $p_e$  and  $N_{min}$ , given  $s = 0.4$  and  $k = 12$ .

0.4 and  $k$  is set to the aforementioned optimal value 12. The three curves in this figure correspond to the setting  $d = \frac{1}{1000}$ ,  $\frac{1}{2000}$ , and  $\frac{1}{5000}$  respectively. For example, when there are 1,000 attackers attacking with the same intensity, to be able to achieve the estimation error of 0.1, the victim needs to receive and use at least 80,000 attack packets. All curves go downward, matching the intuition that larger number of attack packets are needed for traceback when smaller estimation error is desired.

Figure 3 also shows how  $N_{min}$  scales with the number of attackers. We can see that  $N_{min}$  grows almost linearly with the number of attackers for all desired estimation accuracies. For example, when the desired accuracy is 0.1, we need 80,000, 166,000, 450,000 packets for scenarios which have 1,000, 2,000, and 5,000 attackers with the same intensity, respectively. We will show in Section IV that this scaling behavior obtained from information theory matches with scaling behavior obtained from experiments on real-world Internet topologies.

#### IV. PERFORMANCE EVALUATION

We have conducted extensive simulation on three real-world network topologies to evaluate the performance of the proposed scheme, using a simulation tool we have developed. The goal of our simulation is threefold. *First*, we would like to investigate the performance of our traceback scheme. We show that our scheme can achieve high

traceback accuracy even when there are a large number of attackers, and only requires the victim to collect and use a moderate number of attack packets. We also show the performance of our ORMS scheme is orders of magnitude more efficient than the naive approach. *Second*, we are interested in knowing how well our information-theoretic results match with our simulation results. We show that they agree with each other very well. *Third*, we simulate the performance of the proposed scheme in attack scenarios where attackers attack with different rates. The result of simulation shows that our scheme can get good performance with small additional number of packets.

##### A. Simulation set-up: topologies and metrics

The following three real-world network topologies are used in our simulation study.

- Skitter data I – collected from a CAIDA-owned host (a-root.skitter.caida.org) on 11/28/2001 as a part of the Skitter project [1]. This data contains the traceroute data from this server to 192,900 destinations.
- Skitter data II – collected from another CAIDA host (e-root.skitter.caida.org) on 11/27/2001, containing routes to 158,181 destinations.
- Bell-lab’s dataset – collected from a Bell-labs host [5], containing routes to 86,813 destinations. We merged six route sets originated from the same host into one and trimmed incomplete paths.

All three topologies are routes from a single origin to many destinations in the Internet. In our simulation, we assume that this origin is the victim and the attackers are randomly distributed among the destination hosts.

Table 1 shows the performance metrics and control parameters used in our simulation. The accuracy of the traceback scheme is characterized by the sum of FNR and FPR. Among the control parameters,  $N_a$  denotes the number of attackers, and  $N_p$  represents the number of attack packets that are used for traceback. The larger  $N_p$  is, the higher the traceback overhead is. Recall that  $p$  denotes the sampling rate,  $k$  denotes the number of hash functions

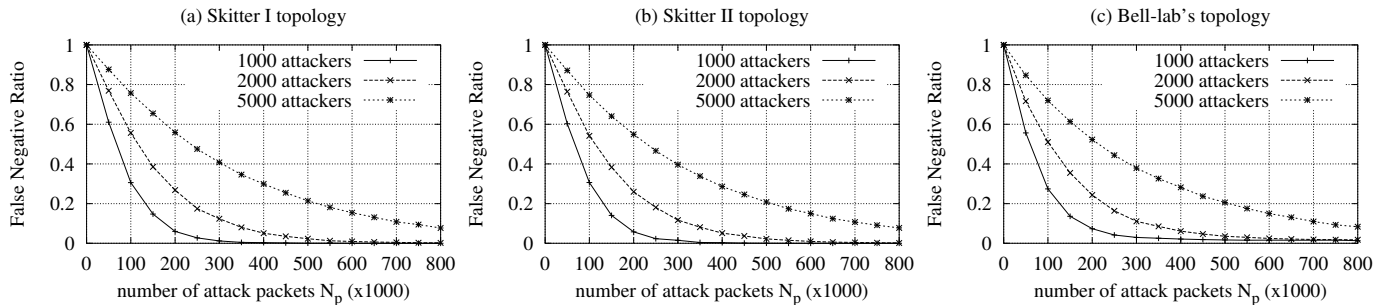


Fig. 4. False Negative Ratio of our traceback scheme on three different topologies

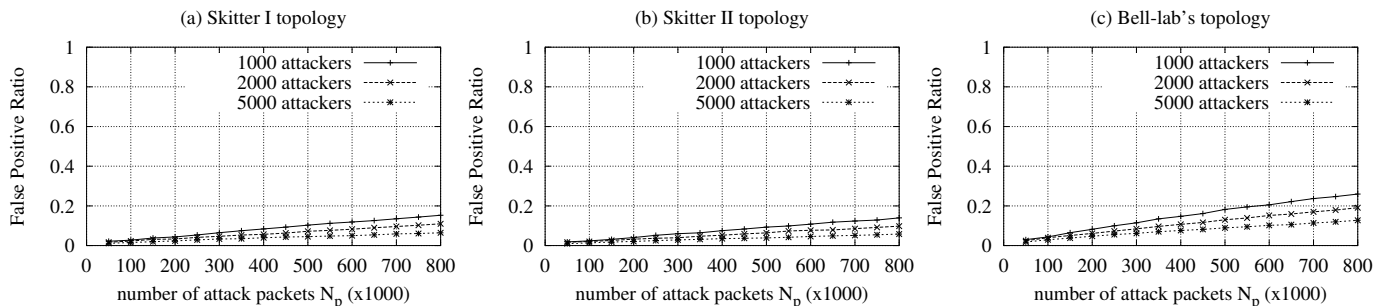


Fig. 5. False Positive Ratio of our traceback scheme on three different topologies

Performance	FNR(False Negative Ratio): the ratio of the number of missed routers in the constructed attack tree to the number of infected routers
Metrics	FPR(False Positive Ratio): the ratio of the number of incorrectly convicted routers to the number of convicted routers in the constructed attack tree
Control	$N_a$ : number of attackers
Parameters	$N_p$ : number of attack packets used for traceback
	$p$ : the sampling rate at an intermediate router
	$k$ : number of hash functions in a Bloom filter
	$s$ : resource constraint ( $=k * p$ )

TABLE I

PERFORMANCE METRICS AND CONTROL PARAMETERS

used for each Bloom filter, and  $s = kp$  is the computational complexity per packet.

In section IV-B.1 and IV-B.2, we first present the results from scenarios in which each attacker attacks the victim with the same rate and the number of packets from each attacker is  $N_p/N_a$ . We then, in section IV-B.3, show simulation results from scenarios where attackers attack with different rates. We assume every router uses the same values of  $s$  and  $p$  for evaluation purpose.

## B. Simulation results

1) *Performance of our scheme and benefit of one-bit marking*: We would like to first investigate how our traceback scheme performs in terms of FPR and FNR with

respect to different number of attackers and different number of total attack packets. Figures 4(a,b,c) shows the FNR of our scheme against the number of attack packets  $N_p$  used for traceback, under the three aforementioned Internet topologies. Similarly, Figures 5(a,b,c) shows the FPR values. In all six figures, we assume  $s = 0.4$  (devote 0.4 bits of computation to each packet). We set  $k$  to 12 bits and  $p$  to 3.3% ( $12 * 3.3\% = 0.4$ ), which corresponds to the optimal parameter setting prescribed by the information-theoretic framework in Section III-C.1. The three curves in each figure correspond to 1,000, 2,000 and 5,000 attackers, respectively.

For all curves in Figures 4(a,b,c), we observe that as the number of attack packets used for traceback  $N_p$  increases, FNR value decreases sharply, which corresponds to more and more infected routers being identified. On the other hand, the FPR value in Figure 5(a,b,c) increases very slowly and is always reasonable. The increase of FPR is caused by our “single-packet decoding rule”. In general, the lower false negative we get from larger  $N_p$  significantly outweighs the slightly higher false positive.

We also observe that our scheme can achieve very high traceback accuracy with a reasonable number of attack packets. For example in Figure 4(a), under the attack from 1,000 attackers, about 175,000 attack packets would be enough to track more than 90% of the infected routers, resulting in only 4.4% FPR. In this case, the average number of packets per attacker is 175. As the number of attackers increases, the number of packets to achieve the same accu-

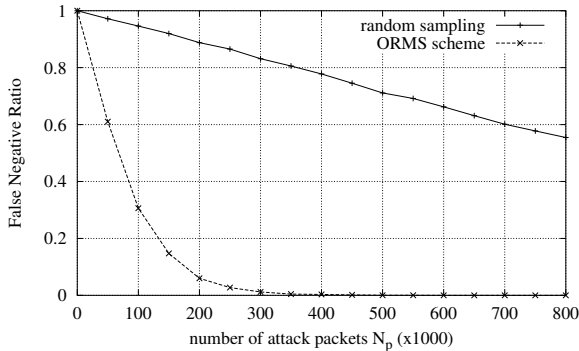


Fig. 6. Benefit of the ORMS scheme

accuracy also increases. However, normalized over the number of attackers, this number actually decreases. For example, to track 90% of the infected routers when there are 2,000 or 5,000 attackers, we need 325,000 or 725,000 packets, respectively. The normalized numbers in these two cases are 160 and 145, respectively. The reason is that, the more attackers there are, the easier it is to identify the infected routers located not too far from the victim.

ORMS scheme significantly outperforms the naive independent sampling approach as demonstrated in Figure 6. Both curves show how the false negatives vary with the number of attack packets used for traceback. Upper and lower curves correspond to the naive scheme and our scheme, respectively. Both are simulated on Skitter I topology when there are 1,000 attacks, and the other parameter settings are the same as before. Clearly, to achieve the same traceback accuracy (false negative), the naive scheme requires orders of magnitude more attack packets for traceback than our scheme. For example, the naive scheme needs 800,000 attack packets to improve the traceback accuracy to 45%, while our scheme can reach an accuracy of 95% with only 200,000 attack packets. As discussed before, the reason behind such an improvement is that our scheme increases the correlation between two routers typically by an order of magnitude.

2) *Verification of theoretical analysis:* In Section III, we have developed an information-theoretic framework for optimal parameter tuning. In particular, we predict that when the resource constraint is  $s = 0.4$ , our knowledge about the attackers is maximized when  $k = 11$  or  $12$  if there are 1,000 attackers with same intensity. We conduct simulations on all topologies to verify the accuracy of our model, and the results are shown in Figure 7(a,b,c). Here the number of attackers  $N_a$  is 1,000. We use the sum of FNR and FPR to represent the overall error level of the simulation results, since the entropy concept captures both FNR and FPR<sup>13</sup>. The three curves correspond to using 50,000, 75,000 and 100,000 attack packets for

<sup>13</sup>The error  $p_e$  does not correspond exactly to FNR + FPR, but is close to FNR + FPR when both numbers are reasonably small.

traceback, respectively. These figures show that the optimal value of  $k$  parameter in our simulation is either 11 or 12, matching our theoretical prediction perfectly. Moreover, we can also see that the shape of these three curves in Figure 7(a) matches those in Figure 2(a) closely. Note that the same parameters are used in both figures. This further demonstrates the close match between theory and practice. This near-perfect match between theory and practice can also be observed in Figure 7(b) and 7(c) for each of the other two topologies respectively.

We also simulate, given a fixed  $k$  value, how the error rate vary with different  $s$  values, and the results are shown in Figure 8(a,b,c). Here the number of attackers  $N_a$  is set to 2,000 and the number of attack packets used for traceback is 200,000. The nine curves in each figure represent the error rates when  $k$  is set to 8, 9,  $\dots$ , and 16, respectively. Among different  $k$  values, our traceback scheme works best with  $k = 12$  when the resource constraint  $s$  is no more than 0.6. When there are more resources (i.e.,  $s > 0.6$ ), our traceback scheme work best with larger  $k$  values. The interpretation of this is that our “one packet decoding rule” generates more false positives when larger  $s$  allows for higher sampling rate and hence larger  $|L_{R_1}|$  (number of attack packets that match the Bloom filter at  $R_1$ ). Since FNR at this point is already low, the increase on the FPR will wipe out the gain we have on FNR. In other words, at this point, the larger  $|L_{R_1}|$  becomes a liability rather than an asset. Therefore, when  $s > 0.6$ , our scheme achieves lower (FNR + FPR), when  $k$  is increased to reduce the false positive ratio of the Bloom filter and the size of  $L_{R_1}$ .

We also would like to compare the minimum number of packets needed to achieve a certain level of traceback accuracy with the theoretical lower bound we have established in section III-C.2. This can be achieved by comparing the curves in each Figure 9(a,b,c) and curves in Figure 3 (in section III-C.2). The parameter settings used in both figures are the same. All three curves in each Figure 9(a,b,c) are higher than curves in Figure 3. In other words, the number of required packet to achieve a certain error rate in the simulation is higher than the number from the theoretical analysis. This is expected for two reasons. First, the error  $p_e$  in the theoretical context is different from (FNR + FPR). In the theoretical context, the error  $p_e$  corresponds to the decoding error when  $R_1$  is correctly convicted and only  $R_2$  is in question. In the (FNR + FNR) measure, however, even  $R_1$  may not have been correctly convicted. Therefore, (FNR + FPR) values are always higher than  $p_e$  values under the same attack scenario. Second, recall that the actual achievable data rate through a channel is always smaller than the Shannon capacity, since the error correction schemes are not perfect. By the same token, our traceback scheme, which can be viewed as a particular “coding/decoding scheme”, will not

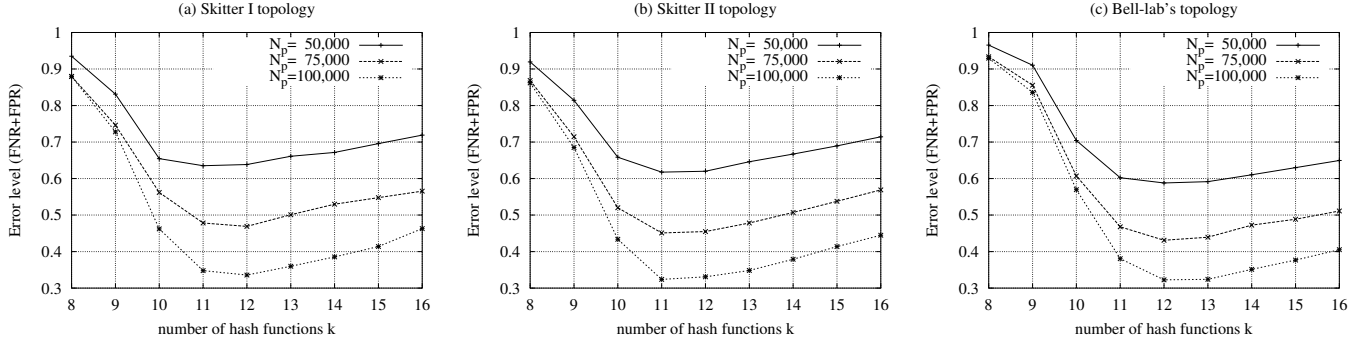


Fig. 7. Simulation results supporting the theoretical analysis : Error level by varying  $k$

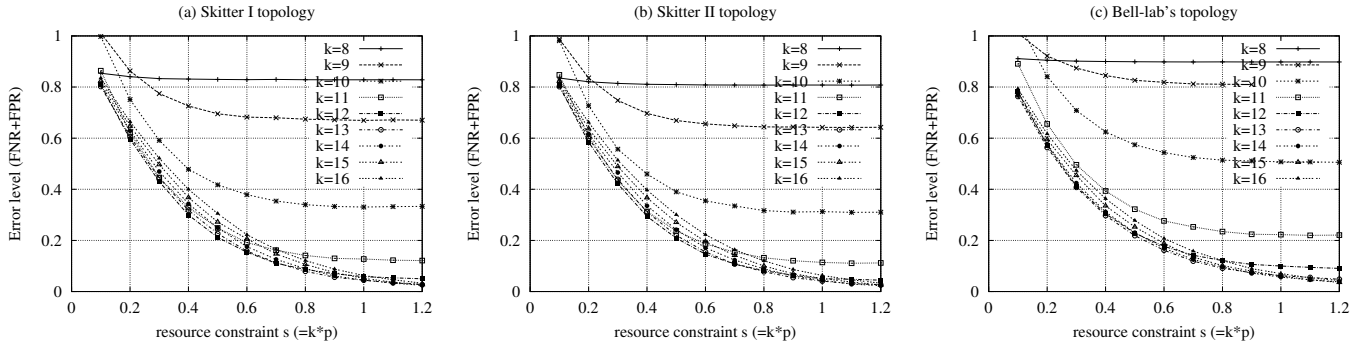


Fig. 8. Simulation results supporting the theoretical analysis : Error level by varying  $s$

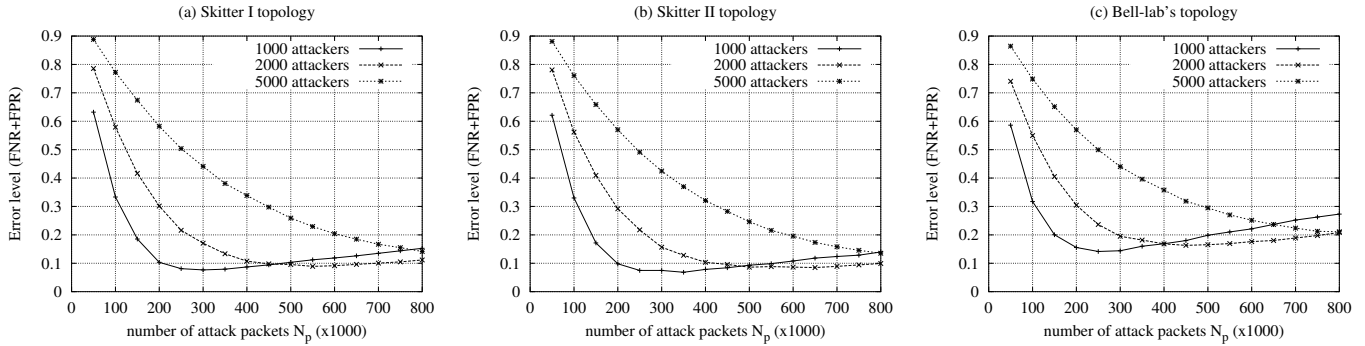


Fig. 9. Simulation results supporting the theoretical analysis : Error level by varying  $N_p$

have the 100% efficiency needed to match its “Shannon capacity” (the lower bound computed from the information theory).

Nevertheless, we observe that the shape of these curves in all figures are similar. In other words, the efficiency of our “coding/decoding scheme” relative to its “Shannon capacity” is insensitive to other system parameters. Note that curves in Figure 9(a,b,c) corresponding to 1,000, and 2,000 attackers goes up when a large number of attack packets ( $N_p$ ) are used for traceback. Our explanation is that when  $N_p$  becomes larger, there are more false positives due to the “one packet decoding rule.” In this case, the decrease in FNR is moderate and is outweighed by the

increase in FPR.

3) *Attack scenario in hybrid bandwidth environment* : During an DDoS attack, the bandwidth of the attackers can be various. Some of the attackers can have more bandwidth than others. To test the proposed scheme in this situation, we assume that half of given number of attackers send five times more traffic than the rest of the attackers (we will call this setting as “hybrid bandwidth environment” hereafter). Figure 10(a,b,c) and Figure 11(a,b,c) show the FNR and FPR, respectively, of the ORMS scheme in the hybrid bandwidth environment. Except the bandwidth of the attackers, all the other parameters are the same as the Figure 4(a,b,c) and Figure 5(a,b,c)

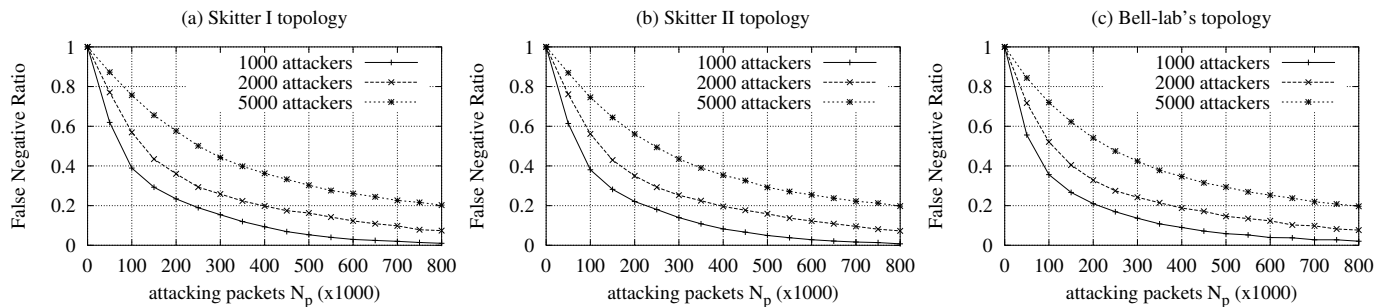


Fig. 10. False Negative Ratio of our traceback scheme in hybrid bandwidth environment

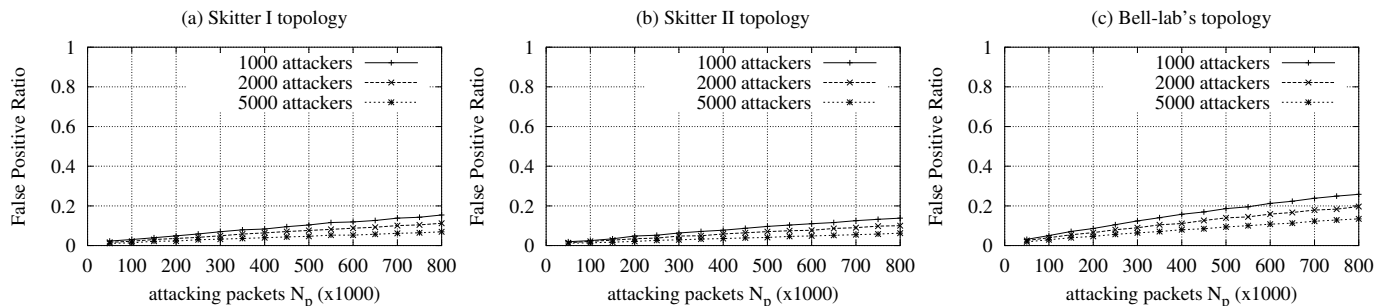


Fig. 11. False Positive Ratio of our traceback scheme in hybrid bandwidth environment

( $p=3.3\%$ ,  $k=12$ ,  $s=0.4$ ,  $N_a=1000, 2000, 5000$ ). All the FNR values are higher than the values in normal bandwidth environment, but still we can get good performance results with small additional number of packets.

## V. RELATED WORK

Recent large-scale DDoS attacks have drawn considerable attention [13]. Two classes of solutions have been proposed to address the problem. One class is the IP traceback schemes [2], [4], [9], [19], [21], [8], [20], [14] that we have discussed in detail in Section I. The second class is the techniques to prevent DDoS attacks and/or to mitigate the effect of such attacks while they are raging on [16], [15], [25], [23], [22], [24], [12], [18]. Prevention mechanisms proactively filter attack packets at strategic places in the network. For example, Ferguson proposes to deploy *Ingress filtering* in routers to detect and drop packets sent using spoofed IP addresses which do not belong to the stub network. Park *et al.* [18] propose to install packet filters at the borders of autonomous systems to filter packets traveling between them. Schemes in both [16] and [25] use router throttles to allocate the victim bandwidth equally (in [16]) or in a min-max fashion ([25]) among perimeter routers. All these schemes aim at filtering out attack traffic or throttling its volume, thereby making legitimate traffic easier to go through.

## VI. CONCLUSION

In this paper, we have presented a new approach to IP traceback based on logging sampled packet digests. In this approach, the sampling rate is low enough for the scheme to scale to very high link speed (e.g., OC-768). To improve the traceback accuracy despite the low sampling rate, we introduce ORMS, a novel one-bit random marking and sampling technique. It significantly increases the correlation between neighboring routers, thereby enabling our traceback scheme to achieve very high traceback accuracy and efficiency. ORMS is also shown to be resistant to the tampering by the attackers. We analyze the proposed scheme based on a novel information-theoretic framework. This framework allows us to compute the parameters with which our system achieves the optimal performance. It also allows us to answer important questions concerning the trade-off between the amount of evidence the victim uses for traceback (the number of attack packets) and the traceback accuracy. Our simulation results show that the proposed scheme performs very well with a reasonable number of attack packets as “evidence”, even when there are thousands of attackers and the sampling rate is as low as 3.3%.

## REFERENCES

- [1] CAIDA's Skitter project web page. Available at <http://www.caida.org/tools/measurement/skitter/>.
- [2] S. Bellovin. Internet draft: ICMP traceback messages. Technical report, Network Working Group, March 2000.

- [3] B. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the Association for Computing Machinery*, 13(7):422–426, 1970.
- [4] H. Burch and B. Cheswick. Tracing anonymous packets to their approximate source. In *Proc. USENIX LISA 2000*, December 2000.
- [5] B. Cheswick. Internet mapping. Available at <http://cm.bell-labs.com/who/ches/map/dbs/index.html>, 1999.
- [6] S. Cohen and Y. Matias. Spectral bloom filters. In *Proc. ACM SIGMOD Conference on Management of Data*, 2003.
- [7] Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. Wiley, 1991.
- [8] D. Dean, M. Franklin, and A. Stubblefield. An algebraic approach to IP traceback. In *Proc. NDSS 2001*, pages 3–12, February 2001.
- [9] T. Doepfner, P. Klein, and A. Koyfman. Using router stamping to identify the source of IP packets. In *Proc. ACM CCS-7*, pages 184–189, November 2000.
- [10] N.G. Duffield and M. Grossglauser. Trajectory sampling for direct traffic observation. *IEEE/ACM Transactions on Networking*, 9(3):280–292, 2000.
- [11] L. Fan, P. Cao, J. Almeida, and A.Z. Broder. Summary cache: A scalable wide-area Web cache sharing protocol. *IEEE/ACM Transactions on Networking*, 8(3):281–293, 2000.
- [12] P. Ferguson. *Network Ingress Filtering: Defeating Denial of Service Attacks Which Employ IP Source Address Spoofing*. RFC 2267, January 1998.
- [13] L. Garber. Denial-of-service attacks rip the Internet. *IEEE Computer*, 33(4):12–17, April 2000.
- [14] Michael T. Goodrich. Efficient packet marking for large-scale IP traceback. In *Proc. of ACM CCS*, November 2002.
- [15] F. Kargl, J. Maier, S. Schlott, and M. Weber. Protecting web servers from distributed denial of service attacks. In *WWW-10*, May 2001.
- [16] R. Mahajan, S. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker. Controlling high bandwidth aggregates in the network. Technical report, ACIRI and AT&T Labs Research, February 2001.
- [17] D. McGuire and B. Krebs. Attack on internet called largest ever. <http://www.washingtonpost.com/wp-dyn/articles/A828-2002Oct22.html>, October 2002.
- [18] K. Park and H. Lee. On the effectiveness of route-based packet filtering for distributed DoS attack prevention in power-law Internets. In *Proc. ACM Sigcomm'2001*, August 2001.
- [19] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Practical network support for IP traceback. In *Proc. ACM SIGCOMM 2000*, pages 295–306, August 2000.
- [20] A. Snoeren, C. Partridge, et al. Hash-based IP traceback. In *Proc. ACM Sigcomm'2001*, August 2001.
- [21] D. Song and A. Perrig. Advanced and authenticated marking schemes for IP traceback. In *Proc. Infocom 2001*, April 2001.
- [22] M. Sung and J. Xu. IP Traceback-based Intelligent Packet Filtering: A Novel Technique for Defending Against Internet DDoS Attacks. In *Proc. of IEEE ICNP'2002*, November 2002.
- [23] J. Xu and W. Lee. Sustaining availability of web services under severe denial of service attacks. *IEEE Transaction on Computers, special issue on Reliable Distributed Systems*, February 2003.
- [24] Abraham Yaar, Adrian Perrig, and Dawn Song. Pi: A path identification mechanism to defend against DDoS attacks. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2003.
- [25] David K.Y. Yau, John C.S. Lui, and Feng Liang. Defending against distributed denial-of-service attacks with max-min fair server-centric router throttles. In *Proc. of IEEE IWQoS*, May 2002.

## APPENDIX

A. Computing  $H(Z|X_{t_1} + X_{f_1}, Y_t + Y_f)$ 

In this appendix, we describe how to compute the condition entropy  $H(Z|X_{t_1} + X_{f_1}, Y_t + Y_f)$  (introduced in Section III-C.1). We will use the same notations as in Section III-C.1.

The number of attack packets  $X_{t_1}$  sampled by router  $R_1$  is a binomial random variable. The probability mass function of  $X_{t_1}$  is  $\Pr(X_{t_1} = k) = \binom{N_p d_1}{k} p^k (1-p)^{N_p d_1 - k}$ . Given a value of  $X_{t_1}$ , the number of false positives  $X_{f_1}$  when all the attack packets are queried against the Bloom filter at router  $R_1$  is a binomial random variable. The probability mass function of  $X_{f_1}$  is given as follows:

$$\Pr(X_{f_1} = k) = \sum_{i=0}^{N_p d_1} \Pr(X_{t_1} = i) \binom{N_p d_1 - i}{k} f^k (1-f)^{N_p d_1 - i - k}$$

Let  $X = X_{t_1} + X_{f_1}$  and  $Y = Y_t + Y_f$ . The probability mass function of  $X$  is given as follows:

$$\Pr(X = k) = \sum_{i=0}^{\min(k, N_p d_1)} \Pr(X_{t_1} = i) \Pr(X_{f_1} = k - i)$$

The probability mass function of the pair of random variables  $(X, Y)$  conditioned on  $Z = 1$  is given as follows:

$$\begin{aligned} \Pr(X = j, Y = i | Z = 1) \\ = \Pr(X = j | Z = 1) \Pr(Y = i | X = j, Z = 1) \end{aligned}$$

The probability mass function of  $\Pr(Y = i | X = j, Z = 1)$  is given as follows:

$$\begin{aligned} \Pr(Y_t + Y_f = i | X = j, Z = 1) &= \sum_{k=0}^{\min(i, N_p d_2)} \Pr(Y_t = k | X = j, Z = 1) \\ &\Pr(Y_f = i - k | X = j, Y_t = k, Z = 1) \end{aligned}$$

where  $\Pr(Y_f = i - k | X = j, Y_t = k, Z = 1) = \binom{j-k}{i-k} f^{i-k} (1-f)^{j-i}$ .

Now all we need is to compute  $\Pr(Y_t = k | X = j, Z = 1)$ . Its computation is a bit involved. We will show how to compute it step by step. The random variable  $X$  (i.e.  $X_{t_1} + X_{f_1}$ ) and  $Y_t$  satisfies  $X_{t_1} = Y_t + W_1 + W_2$  where  $W_1$  has the probability distribution  $\text{Binom}(N_p d_2 - X_{t_2}, p / (2 - p))$  and  $W_2$  is a binomial random variable with parameters  $(N_p d_1 - N_p d_2, p)$ . Intuitively, the attack packets that  $R_1$  sampled composes of three parts: (1)  $Y_t$  number of attack packets that  $R_2$  has sampled; (2)  $W_1$  number of attack packets sampled from the set of attack packets that are not sampled by  $R_2$ ; (3)  $W_2$  number of attack packets sampled from attack packets coming from upstream neighbors other than  $R_2$ . We assume  $d_1$  is equal to  $d_2$ , because it captures the “common case” as explained in Section III-C.1. Since  $\Pr(Y_t = k | X = j, Z = 1) = \sum_{l=0}^j \Pr(X_{f_1} =$

$l|Z = 1)\Pr(Y_t = k|X_{t_1} = j - l, X_{f_1} = l, Z = 1)$ , all we need to calculate is  $\Pr(Y_t = k|X_{t_1} = j - l, X_{f_1} = l, Z = 1)$ . It is given as follows:

$$\begin{aligned}
& \Pr(Y_t = k|X_{t_1} = j - l, X_{f_1} = l, Z = 1) \\
= & \sum_{g=k}^{N_p d_2} \Pr(X_{t_2} = g|Z = 1) \cdot \\
& \Pr(Y_t = k, w_1 = j - l - k|X_{t_1} = j - l, X_{f_1} = l, X_{t_2} = g, Z = 1) \\
= & \sum_{g=k}^{N_p d_2} \Pr(X_{t_2} = g|Z = 1) \cdot \\
& \Pr(Y_t = k|X_{t_2} = g, Z = 1)\Pr(w_1 = j - l - k|X_{t_2} = g, Z = 1) \\
= & \sum_{g=k}^{N_p d_2} \binom{N_p d_2}{g} p^g (1-p)^{(N_p d_2 - g)} \cdot \binom{g}{k} \left(\frac{1}{2-p}\right)^k \left(\frac{1-p}{2-p}\right)^{(g-k)} \cdot \\
& \binom{N_p d_2 - g}{j - l - k} (p/(2-p))^{j-l-k} (1-p/(2-p))^{N_p d_2 - g - j + l + k}
\end{aligned}$$

Once we have computed the  $\Pr(X = i, Y = j|Z = 1)$ , then according to formula (2) in Section III-B the conditional entropy can be calculated as follows:

$$\begin{aligned}
& H(Z|X, Y) \\
= & - \sum_{(X, Y)} \Pr(X = i, Y = j, Z = 1) \log_2 \frac{\Pr(X = i, Y = j, Z = 1)}{\Pr(X = i, Y = j)} \\
& - \sum_{(X, Y)} \Pr(X = i, Y = j, Z = 0) \log_2 \frac{\Pr(X = i, Y = j, Z = 0)}{\Pr(X = i, Y = j)}
\end{aligned}$$

where

$$\begin{aligned}
\Pr(X = i, Y = j|Z = 0) &= \Pr(X = i|Z = 0)\Pr(Y = j|X = i, Z = 0) \\
&= \Pr(X = i) \binom{i}{j} f^j (1-f)^{i-j}
\end{aligned}$$

and

$$\begin{aligned}
\Pr(X = i, Y = j) &= \Pr(Z = 0)\Pr(X = i, Y = j|Z = 0) \\
&\quad + \Pr(Z = 1)\Pr(X = i, Y = j|Z = 1) \\
&= \Pr(X_{t_2} = 0)\Pr(X = i, Y = j|Z = 0) \\
&\quad + \Pr(X_{t_2} > 0)\Pr(X = i, Y = j|Z = 1).
\end{aligned}$$