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

Minho Sung, Jun (Jim) Xu, *Member, IEEE,* Jun Li, and Li (Erran) Li, *Member, IEEE,*

***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.* [1], thereby being able to scalable to much higher link speed (e.g., OC-768). The baseline idea of our approach is to sample and log a small percentage (e.g., 3.3%) of packets. The challenge of this low sampling rate is that much more sophisticated techniques need to be used for traceback. Our solution is to construct the attack tree using the correlation between the attack packets sampled by neighboring routers. The scheme using naive independent random sampling does not perform well due to the low correlation between the packets sampled by neighboring routers. We invent a sampling scheme that improves this correlation and the overall efficiency significantly. Another major contribution of this work is that we introduce a novel information-theoretic framework for our traceback scheme 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) match 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*—Information theory, IP traceback, Distributed Denial-of-Service attacks, Network security.**

## I. Introduction

DISTRIBUTED Denial of Service (DDoS) attacks against high-profile web sites such as Yahoo, CNN, Amazon and E*Trade in early 2000 [2] rendered the services of these web sites unavailable for hours or even days. New instances of DDoS attacks continue to be reported. For example, a recent DDoS attack brought down eight root DNS servers in an effort to paralyze the Internet [3]. It is clear that DDoS attacks will not stop or scale down until they are properly addressed.

A preliminary version of the paper was presented at IEEE Symposium on Security and Privacy, held in Oakland, California, USA, from May 9th to 12th, 2004. Since the presentation, we have performed additional research into the problem, which results in a new section (Section VII) in this manuscript.

This work is supported in part by NSF grants CNS 0716423, CNS 0626979, CNS 0519745, and NSF CAREER award CNS 0238315.

Minho Sung was with the College of Computing, Georgia Institute of Technology, Atlanta, GA 30332 USA. This work was performed while he was with Georgia Tech. He is now with Telecommunication R&D Center, Samsung Electronics Company, Ltd. Suwon, Korea. (e-mail: m.sung@samsung.com)

Jun (Jim) Xu and Jun Li are with the College of Computing, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: {jx,junli}@cc.gatech.edu).

Li (Erran) Li is with Bell Labs, Lucent Technologies (e-mail: erranlli@bell-labs.com).

One possible way to counter DDoS attacks is to trace the attack sources 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 does not keep track of the path traversed by a packet. 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 [4], [5], [6], [7], [8]. By receiving a significant number of packets, the victim can construct the attack paths. This is referred to as the Probabilistic Packet Marking (PPM) scheme.
2) The other is to store packet digests in the form of Bloom filters [9] at each router [1]. By checking neighboring routers iteratively with attack packets, the attack path of a flow can be constructed. This is referred to as the hash-based scheme.

However, both traceback schemes suffer from scalability problems. As we will show in the next section, PPM schemes cannot scale to large number of attackers. The best scheme proposed can only efficiently trace fewer than 100 attackers using a 17-bit marking field (discussed later). The hash-based scheme is not scalable for high-speed links since recording all 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 the 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. A recent PPM scheme proposed by Goodrich [8] is shown to be the most scalable[1] among the PPM schemes. However, with a marking field of 17 bits, it can only scale up

[1]Song *et at.*'s scheme [6] allows for 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 to authenticate the victim, which can be complicated to deploy and administer.

to attack trees containing 100 routers[2]. 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 traceback.

Hash-based approach, on the other hand, is very effective for large-scale IP traceback since it needs only a single packet to trace one attacker [1]. However, since it computes and stores a Bloom filter digest for every packet, its computational and storage overhead is prohibitive for a router operating on very high speed links. For example, assuming a 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 44GB of storage space every hour, with the parameters suggested in [1]. 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. New contributions

Our technical contributions are two-fold. First, we propose a novel packet logging based traceback scheme that is scalable to high link speeds. The basic idea of our approach is to sample a small percentage (e.g., 3.3%) of packets. We construct the attack tree using the correlation between the attack packets sampled by neighboring routers. The scheme with naive independent random sampling does not perform well due to the low correlation between the packets sampled by neighboring routers. We invent a sampling scheme that improves this correlation and the overall efficiency by several orders of magnitude. Sampling greatly reduces the computational and storage overhead for packet logging. For example, with a sampling rate of 3.3% (it can be smaller), our storage overhead is only $0.4/\ln 2$ bits per packet[3]. A duplex OC-192 link will require a computation of 8 million hash functions every second and a storage of 5.2GB for one hour's traffic. This is an order of magnitude more affordable than the scheme in [1].

Our second major contribution is to introduce a novel information-theoretic framework for our traceback scheme to answer important questions on system parameter tuning and on the fundamental trade-off between the resource used for traceback and the traceback accuracy. For a given performance constraint, there is the question of how to tune the traceback scheme in terms of the number of hash functions and the sampling rate. This optimization problem is formulated as a channel capacity maximization problem in information theory. This framework also allows us to compute the minimum number of attack packets needed for achieving certain traceback accuracy and to study how this number scales to larger number of attackers.

Our 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 locate 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 for traceback (resulting in a query size of 4.2MB[4]), even when there are 1,000 attackers (175 packets from an attacker in average).

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

## II. Related Work

Recent large-scale DDoS attacks have drawn considerable attention [2]. The broad research efforts on defending DDoS attacks can be classified into three categories.

**1. Attack detection and classification.** Many techniques have been proposed to detect ongoing DDoS attacks, which can be classified into either signature-based (e.g., [10]) or statistics-based (e.g., [11]). As we have mentioned, these attack detection techniques are needed to trigger our traceback procedure. Hussain *et al.* [12] propose a framework to classify DoS attacks into single source or multiple sources. This classification information can help the victim to respond effectively to the attacks.

**2. Attack response mechanisms.** Two classes of solutions have been proposed to address the problem. One class is the IP traceback schemes [13], [4], [5], [6], [7], [1], [8], [14] that we have discussed in detail in Section I, including this work. In addition to proposing some PPM-based IP traceback schemes, Adler [14] studied the fundamental tradeoffs between the number of packets needed for traceback and the bits available for performing packet marking, in the PPM context. In this paper, we studied a similar tradeoff question in the context of logging-based IP traceback (i.e., hash-based) and sampling. The techniques used in [14] to derive these two tradeoffs are very different. While techniques in [14] come mostly from theoretical computer science, ours come mostly from information theory. Finally, we find it extremely hard to study this tradeoff question when the network allows both PPM and logging, since the question can be cast as a *network information theory* (mostly unsolved [15]) problem.

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], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27]. In one of our prior work [20], we present a technique that can effectively filter out the majority of DDoS traffic, thus improving the overall throughput of the legitimate traffic. Another prior work of ours [19] proposes a practical

[2]We assume that the "message size" (defined in [8]) is 64 bits for representing the IP addresses of the current router and the previous router, and the "collision size" (defined in [8]) is no more than 2.

[3]Each Bloom filter digest uses 12 hash functions. The reason why we use 12 will be clear in Section V. The term $\ln 2$ is due to the Bloom filter space-efficiency trade-off and will be explained in Section IV-A3.

[4]Only the invariant parts of IP header(16 bytes) and first 8 bytes of the payload will be used for traceback as in [1].

DDoS defense system that can protect the availability of web services during severe DDoS attacks. These two pieces of work fall into the second class. SOS [24] uses overlay techniques with selective re-routing to prevent large flooding attacks. Mitigation mechanisms proactively filter attack packets at strategic places in the network. For example, Ferguson [22] 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.* [23] propose to install packet filters at the borders of autonomous systems to filter packets traveling between them. Yarr *et al.* [21] propose to encode the paths traversed by the packets and filter out the attack traffic according to the path identifier. Jin *et al.* [28] propose to use the TTL values to detect and filter out spoofed IP packets. Schemes in both [16] and [18] use router throttles to allocate the victim bandwidth equally ([16]) or in a min-max fashion ([18]) among perimeter routers. All these schemes aim at filtering out attack traffic or throttling its volume, thereby making legitimate traffic easier to go through.

**3. Understanding DoS attack prevalence and attack dynamics.** Moore *et al.* used "backscatter analysis" to gauge the level of Internet DoS activity [29]. They studied the intensity and duration of the DoS attacks and observed a small number of long attacks constituting a significant fraction of the overall attack volume. Paxson [30] analyzed the reflector attacks that conventional PPM schemes cannot work against. He then proposed a solution called Reflective Probabilistic Packet Marking Scheme (RPPM).

## III. Overview

### A. Our solution for 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 speed. Like [1], our scheme requires Internet routers to record Bloom filter digests of packets going through them. However, unlike [1], which records all of packets, 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 speed or higher rates. For example, our scheme can scale to OC-768 speed (simplex) using only DRAM, when sampling 3.3% of the traffic.

The trade-off of sampling is that it makes the traceback process much more difficult, especially with a low sampling 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 the attack 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. The attack tree starts with the victim as the root and the only leaf. It grows when a leaf node determines that one or more of its neighbors are highly likely to be on an attack path (called "infected" hereafter). Such a likelihood is assessed by performing the following 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 neighbors $R_2$ is likely to be on an 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 test is to check whether "what $R_1$ has seen" has non-negligible 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, $R_2$ will further test its neighbors recursively using this procedure. Designing the aforementioned threshold decoding rule is nontrivial, and careful game-theoretic study is needed to make sure that the rule is loophole-free to the attackers.

Clearly, the higher the correlation between the attack packets sampled by neighboring infected routers, the more accurate our traceback scheme. Given 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 sampled only matches 3.3% of what $R_2$ has sampled. While consistent sampling techniques such as trajectory sampling [31] has the potential to improve this factor by nearly 100%, it will not work for an adversarial environment, as we will discuss in Section IV-A1. 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. Using this technique, our scheme requires a much smaller number of attack packets for traceback, and achieves better traceback accuracy than independent sampling. We can further improve the accuracy of our traceback scheme by using more than one bit for coordination. This is discussed in detail in Section VII.

### B. Information-theoretic framework of our traceback scheme

The design of the scheme leads to a very interesting optimization problem, which can be solved using an information-theoretic framework. We assume that the average number of bits devoted for each packet is a fixed constant $s$, due to the computational and storage constraints of a router. In other words, in average, for each packet we compute $s$ hash functions. Then the number of hash functions 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 hash computations to be performed for each packet, one possible combination is that the router samples 5% of the packets and the number of hash functions is 8 ($5\% \times 8 = 0.4$). With the same resource constraint, an alternative combination is to sample 2.5% of the packets, but the number of hash functions is 16. Which one is better? Intuitively, a higher sampling rate increases the aforementioned correlation between the packets sampled by two routers, making traceback easier. However, the number of hash functions would have to be proportionately smaller, which results in a higher false positive rate in Bloom filter. This adds noise to the aforementioned traceback 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)? By viewing the traceback system as a communication channel, we show that this question can be answered using information theory techniques. The optimal parameter setting should maximize the Shannon capacity of this channel. Our simulation results show that the information-theoretic framework indeed allows us to find the optimal parameter setting.

Our information-theoretic framework also allows us to answer another important question concerning the trade-off between the amount of evidence the victim has to gather (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 a larger number of attackers.

## IV. 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 Sections IV-A and IV-B, we describe the sampling algorithm and the traceback algorithm in detail.

### A. Sampling

In this section, we first explain the challenge in designing the sampling method for traceback. Then we present the details of our proposed sampling method. Finally, for completeness, we review the packet digesting method [1] for hash-based traceback.

*1) A design challenge.:* Our proposed scheme significantly reduces the processing and storage requirements by sampling. However, sampling makes traceback more difficult. In particular, it is now almost impossible to trace one attacker with only one packet as in [1]. This is because, with a low sampling percentage, the first router on the attack path that will sample a particular attack packet is, in 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. 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[5] to trace one attacker. Recall from Section III-A that if a convicted router $R_1$ wants to check whether one of its neighbors $R_2$ is infected, the scheme checks whether the set of packets "$R_1$ has seen" has a non-negligible correlation with the set of packets "$R_2$ has seen". 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 the packets sampled by 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 by over 50% with the same sampling rate $p$ at every router, therefore reaching the $O(\frac{1}{p})$ asymptotic lower bound. We will describe this scheme in the next section.

**Sampling procedure at router $R$**
**(given sampling rate $p$):**
**for** each packet $w$
  **if** ($w$.mark = 1) **then**
    write 0 into $w$.mark;
    store the digest of $w$, subject to a cap of $\frac{p}{2}$;
  **else**
    with probability $\frac{p}{2-p}$
      store the digest of $w$;
      write 1 into $w$.mark;
**if** (marking percentage is not $\frac{p}{2}$) **then**
  tune it to $\frac{p}{2}$;
  /* make the process "stationary" */

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

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* [31]). 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. We explored along this direction and found that it is extremely challenging to design noncryptographic[6] 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.

*2) One-bit Random Marking and Sampling (ORMS).:* Independent random sampling method does not work well since the correlation factor between the packets sampled by neighboring routers is only $p$, the sampling rate. In this section, we present our sampling scheme that significantly improves this correlation factor. The key idea of our scheme is that, besides sampling the packets, a router also 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 use a marking field of only one bit for this coordination (we return to the case of multiple bits in Section VII). This bit can be easily fit into many possible locations in the IP header (e.g., IP fragmentation field [7] ).

[5] Note that $O(\frac{1}{p^2})$ can be orders of magnitude larger than $O(\frac{1}{p})$ when $p$ is small.

[6] Cryptographic techniques may involve key distribution and management to hundreds of thousands of Internet routers.

[7] The IP fragmentation field has been reused in the PPM-based IP traceback schemes. The "backward compatibility" issues has been discussed in [5].

Our ORMS scheme is presented in Figure 1. This algorithm is executed at every interface of the participating routers. 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 proportion 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 $\frac{p}{2-p}$ (where $\frac{p}{2-p}$ comes from will become clear after next paragraph). A router will also measure the proportion of packets coming from itself that are marked. If this proportion is larger or smaller than $\frac{p}{2}$, the router will adjust it to $\frac{p}{2}$ by marking and unmarking some bits (lines 9 & 10 in Figure 1). This can be achieved using traditional rate-control techniques in networking such as leaky bucket [32].

Consider the path from a remote host to the victim. We will show that the two invariants hold in the approximate sense. We assume only the first hop (a router) from the host have other hosts attached to it and all later hops (routers) are neighboring with other participating routers only. The first invariant is that approximately $\frac{p}{2}$ 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 almost (subject to a small error $\epsilon$) do not need to be executed at later routers. To see this, note that at later routers, approximately $(1-\frac{p}{2})$ of the arriving packets are not marked, and among those $\frac{p}{2-p}(1-\frac{p}{2}) = \frac{p}{2-p}\cdot\frac{2-p}{2} = \frac{p}{2}$ will be marked. Therefore, once the system is jump-started to stationarity (with $\frac{p}{2}$ marked), it remains stationary. The second invariant is that each router, except for the first hop (which may sample less than $p$), will sample approximately $p$ 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 $\frac{p}{2}$ and at most $p$ 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 $\frac{p}{2}$ of packets $R_2$ has marked, and among another $\frac{p}{2}$ 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 several orders of magnitude improvement compared to 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). As discussed before, the correlation between neighboring routers will always be $\frac{1}{2-p}$.

*3) Packet digesting.:* Like in [1], we use a space-efficient data structure known as Bloom filter [9] to record packet digests[8].

A Bloom filter representing a set of packets $S = \{x_1, x_2, \cdots, x_n\}$ of size $n$ is described by an array $A$ of $m$ bits, initialized to 0. A Bloom filter uses $k$ independent hash functions $h_1, h_2, \cdots, h_k$ with range $\{1, \cdots, m\}$. During *insertion*, given a packet $x$ to be inserted into a set $S$, the bits $A[h_i(x)]$, $i = 1, 2, \cdots, k$, are set to 1. To *query* for a packet $y$, i.e., to check if $y$ is in $S$, we check the values of the bits $A[h_i(y)]$, $i = 1, 2, \cdots, k$. 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 packet. However, it may contain false positives, i.e., returning "yes" while the packet is not in the set. The *capacity factor*, denoted as $c$, 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 before $c$ decreases to $k/\ln 2$. Then according to [9], the false positive rate of the Bloom filter is no more than $2^{-k}$. In Sections V and VI, the false positive rate of the Bloom filter is always assumed to be $2^{-k}$ for analytical and performance evaluation purposes.

Note that same as in [1], we use the first 24 invariant bytes of an IP packet as the hash input. These 24 bytes include the invariant portion of the IP header (16 bytes) 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 24 invariant bytes.

### B. Traceback processing

When the victim detects a DDoS attack, it will trigger a traceback procedure. The victim will first collect a decent number of attack packets, which is not difficult during a DDoS attack. Then it will use these packets to track down the attackers. We denote the set of packets that is used for traceback as $L_v$, as described in Section I-B. 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 $S$ using the entire 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 match the Bloom filter of $S$ will be assembled into a set $L_S$. Each neighbor $R$ of $S$ will then be queried by $L_S$ (not $L_v$!), if $R$ has not yet been convicted. Again, if at least one

[8] We assume that transformation of the packets such as packet encapsulation can be handled using the transformation lookup table suggested in [1]. Also, we use the same strategies suggested in [1] to fill up the Bloom filters and to store the values of Bloom filter in local storages of the routers after each pre-defined epoch. However, the parameters of the operations are not the same.

match is found, $S$ convicts $R$ and sends $L_v$ to $R$; Otherwise, nothing needs to be done to $R$ by $S$. If $R$ is convicted, $R$ will assemble $L_R$, which is the set of packets in $L_v$ that match the Bloom filter at $R$. The set $L_R$ will then be used by $R$ to query its neighbors. This process is repeated recursively until it cannot proceed.

We now discuss the subtle features of 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 "1" 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[9]. This loophole is closed under our "one-packet decoding rule". It can be easily verified that an attacker has no incentive to send identical packets anymore from a game-theoretic 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 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 $|L_v| >> 5,000$, 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 number of false positives caused by $L_R$ is also much smaller.

## V. 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 V-A. After briefly introducing the relevant information theory concepts and theorems in Section V-B, we show how they are applied to our context in Section V-C.

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

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

*1) Parameter tuning:* We have discussed in Section III-B 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) Tradeoff between traceback overhead and accuracy:* 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 and conditional entropy. Then we introduce Fano's inequality [15], which will be used to answer the question raised in Section V-A2.

*1) Entropy and conditional entropy.:*

*Definition 5.1:* The entropy of a discrete random variable $X$ is defined as

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

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 5.2:* The conditional entropy of a random variable $X$ conditioned on another random variable $Y$ is defined as

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

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 cannot 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 $Y$.

*2) Fano's inequality.:* In our analysis, we would like to estimate the value of $X$ based on the observation of $Y$. 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, the more accurate the estimation. This intuition is captured by Fano's inequality [15].

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 states the following.

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

Here, $H(p_e)$ is "overloaded" to stand for the entropy of the indicator random variable $1_{\{\hat{X}\neq X\}}$. By (1), $H(p_e) =$

[9]One can also use counting Bloom filter [33] or Spectrum Bloom filter [34] 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-theoretic analysis associated with using the higher threshold is extremely complex.

$-p_e \log_2 p_e - (1-p_e)\log_2(1-p_e)$. 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:

$$H(p_e) \geq H(X|Y) \tag{4}$$

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 a wrong result of more than half of the time, we can simply use $\overline{A}$). 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 of 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) Modeling.:* As described in Section IV-B, when a (convicted) router $R_1$ would like to check whether one of its 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., $L_v$).

We first define some notations:

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

In the following, we introduce step by step the random variables involved in the analysis. By convention, $Binom(\mathcal{N}, \mathcal{P})$ represents the binomial distribution with constant parameters $\mathcal{N}$ and $\mathcal{P}$, where $\mathcal{N}$ is the number of trials and $\mathcal{P}$ is the "success" probability. In some places below, we modify the $Binom$ notation slightly to put a random variable in the place of $\mathcal{N}$, which will be made mathematically rigorous as following. 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 modification 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 $L_v$ is queried against the Bloom filter at $R_1$. Its probability distribution is $Binom(N_p - X_{t_1}, f)$. Here $f$ 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 matches instead of Bloom filter false positives) when the Bloom filter at $R_2$ is queried with $L_{R_1}$. Its probability distribution is $Binom(X_{t_2}, \frac{1}{2-p})$. The parameter $\frac{1}{2-p}$ comes from the fact that the correlation factor between the packets sampled by 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 $Binom(X_{t_1} + X_{f_1} - Y_t, f)$.

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.

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

From information theory, the accuracy of estimating $Z$ from observing $X_{t_1}+X_{f_1}$ and $Y_t+Y_f$ is measured by 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. The 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 computing $H(Z|X_{t_1} + X_{f_1}, Y_t + Y_f)$, 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 also assume pr$[Z = 1]$ = pr$[Z = 1] = 1/2$, that is, we assume no prior knowledge about $Z$.

*2) Parameter tuning.:* As we discussed before, our resource constraint is $kp \leq s$. Here $s$ is the number of bits of computation (i.e., the number of hashing operations) devoted to each packet in 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 seen which $k$ value will allow us to determine, with best accuracy, whether $R_2$ has been infected.

By information theory, our knowledge about $Z$ from observing $X_{t_1}+X_{f_1}$ and $Y_t+Y_f$ is maximized when the conditional entropy $H(Z|X_{t_1}+X_{f_1}, Y_t+Y_f)$ is minimized. In other words, we would like to compute

$$k^* = \underset{k}{argmin}\ H(Z|X_{t_1} + X_{f_1}, Y_t + Y_f) \tag{5}$$

subject to the constraint $kp = s$ as discussed before.

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 $d_2$ (we assume $d_1 = d_2$). 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 in estimating the value of $Z$ for those potential $R_2$'s that have the concentration $d$. One may wonder if we target a certain concentration, but

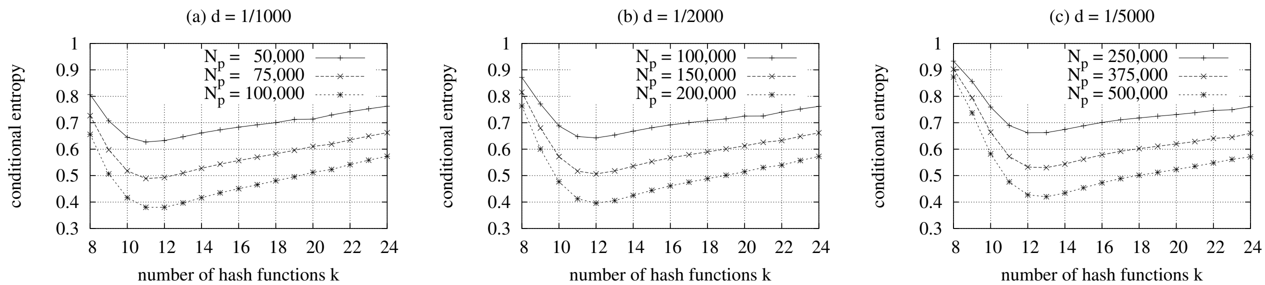


Fig. 2. Conditional entropy with respect to the number of hash functions used in a Bloom filter for $s = 0.4$ with different concentrations

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 other words, the optimality of $k$ is not sensitive to the concentration value we are targeting. Therefore, we can choose a $k$ for our scheme to work well even if we do not know the accurate information of $d_1, d_2$. All we need is the range of $d_1, d_2$.

We illustrate 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 VI, our scheme will adopt $k = 12$ when its resource constraint is $s = 0.4$. Simulation results 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.

*3) 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

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

where $p_e = \Pr[\hat{Z} \neq Z]$ is the probability that our estimation $\hat{Z}$ is different from the actual value of $Z$. 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 $H(\epsilon)$.

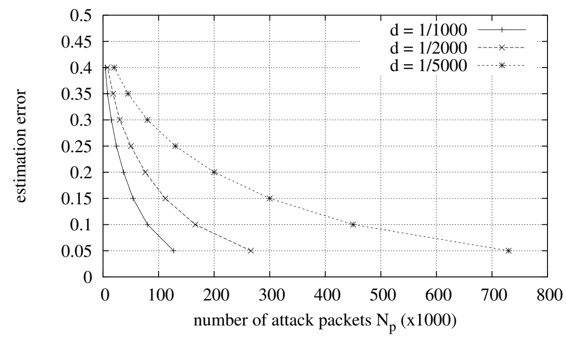


Fig. 3. The trade-off between the estimation error $p_e$ and $N_{min}$, given $s = 0.4$ and $k = 12$.

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

## VI. 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 two-fold. *First*, 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. *Second*, 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.

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

TABLE I
PERFORMANCE METRICS AND CONTROL PARAMETERS

### 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 [35]. 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 lab's host [36], 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 [10].

Table 1 shows the performance metrics and control parameters used in our simulation. 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 the traceback scheme is defined as the ratio of the number of false negatives to the number of infected routers during the attack[11]. Because Bloom filters are used to store packet digests, the traceback system may identify routers that are not actually on attack paths. We call these routers *false positives*. The *false positive ratio* (FPR) of an attack tree constructed by the 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 the services being attacked. Incomplete or approximate attack path information is valuable because the efficiency of complementary measures such as packet filtering improves as they are applied further from the victim and closer to the attack sources [5]. This is why we count routers instead of routes in these performance metrics.

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. A larger $N_p$ leads to a higher traceback overhead. Recall that $p$ denotes the sampling rate, $k$ denotes the number of hash functions used for each Bloom filter, and $s = kp$ is the computational complexity per packet. We assume every router uses the same values of $s$ and $p$ for evaluation purpose. For the purpose of simulations, we also assume all the intermediate routers do the marking and store the packet digests.

### B. Verification of theoretical analysis

In Section V, we have developed an information-theoretic framework for optimal parameter tuning. In particular, we predict that when the resource constraint $s = 0.4$, the traceback accuracy 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 Figures 4(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 reflects both FNR and FPR[12]. The three curves correspond to using 50,000, 75,000 and 100,000 attack packets for traceback, respectively. These figures show that the optimal value of $k$ parameter in our simulation is either 11 or 12, matching our theoretical prediction very well.

For example, when we use 12 hash functions in a Bloom filter and use 100,000 attack packets for traceback on Skitter I topology, we can get 0.308 and 0.027 as FNR and FPR respectively. It means that we can correctly identify around 70% of infected routers in the attack tree with only 2.7% of false positive. Note that this result is obtained using very low resource constraint $s = 0.4$ which allows the sampling rate to be as low as 3.3%.

We also simulate, given a fixed $k$ value, the error rate versus different $s$ values, and the results are shown in Figures 5(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 $8, 9, \cdots$, and 16, respectively. Among the different $k$ values, our traceback scheme performs best with $k = 12$ when the resource constraint $s$ is no more than 0.6. For example, when $k = 12$ and $s = 0.6$, we get 0.009 and 0.061 as FNR and FPR respectively. When there are more resources (i.e., $s > 0.6$), our traceback scheme performs better 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

[10] In real situation, this assumption of random distribution can be wrong because many hosts in same vulnerable network can be compromised simultaneously. However, clustering of attackers only helps our scheme because it increases the correlation between routers in attack path.

[11] Recall that a router on the attack path of an attacker is called "infected".

[12] The error $p_e$ does not correspond exactly to FNR + FPR, but is close to FNR + FPR when both numbers are reasonably small.

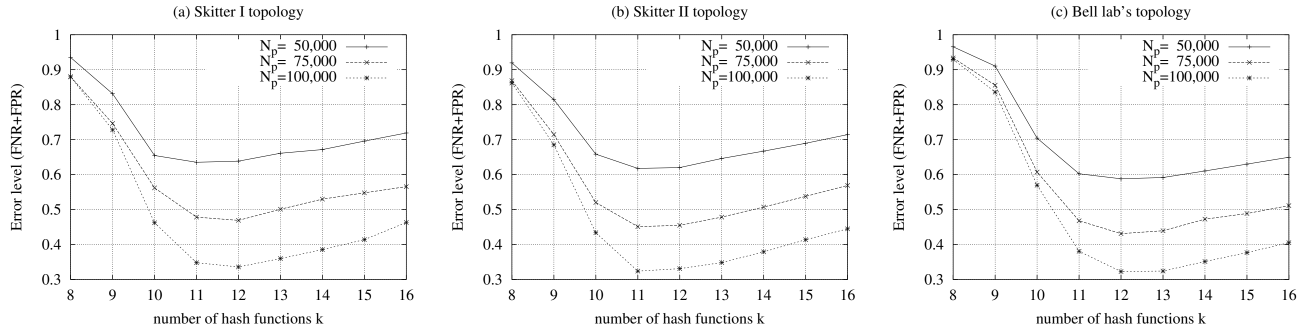


Fig. 4. Simulation results supporting the theoretical analysis (error level by varying $k$)

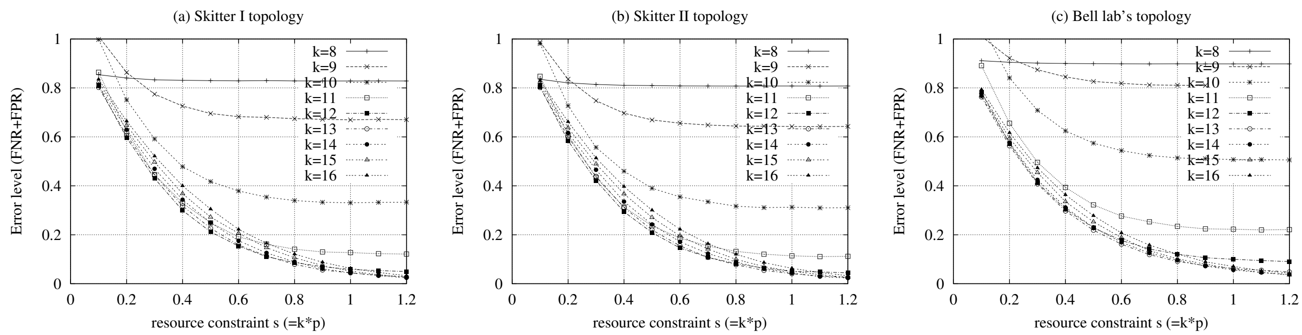


Fig. 5. Simulation results supporting the theoretical analysis (error level by varying $s$)

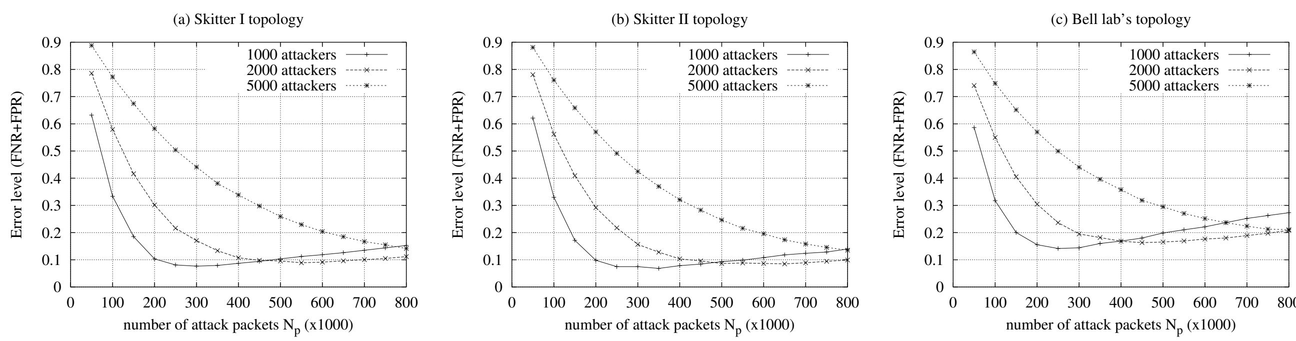


Fig. 6. Simulation results supporting the theoretical analysis (error level by varying $N_p$)

$|L_{R_1}|$ 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 rate 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 V-C3. This can be achieved by comparing the curves in Figures 6(a,b,c) and curves in Figure 3 (in Section V-C3). The parameter settings used in all figures are the same. All three curves in each figure of Figures 6(a,b,c) are higher than curves in Figure 3. In other words, the required number of packets to achieve a certain error rate in the simulation is higher than the number from the theoretical analysis. This is expected for the following reason. 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. Note that curves in Figures 6(a,b,c) corresponding to 1,000 and 2,000 attackers go up when a large number of attack packets 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 outweighs the increase in FPR.

## C. Performance of our scheme

We would like to investigate how our traceback scheme performs in terms of FPR and FNR with respect to different

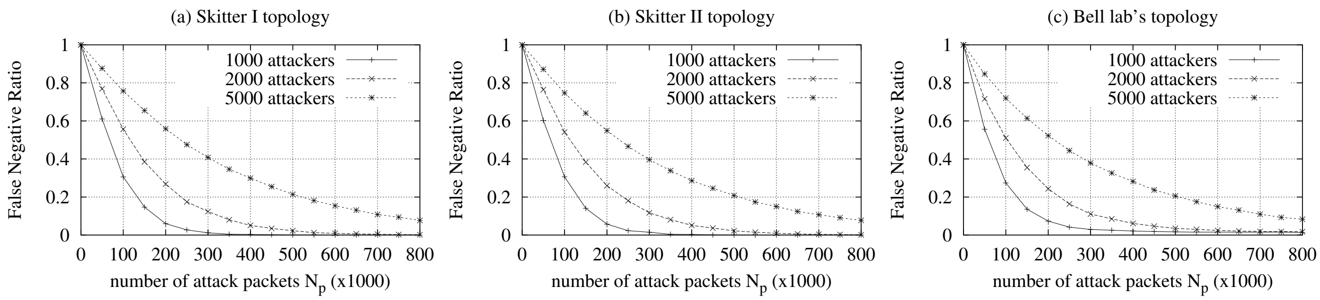


Fig. 7. False Negative Ratio of our traceback scheme on three different topologies

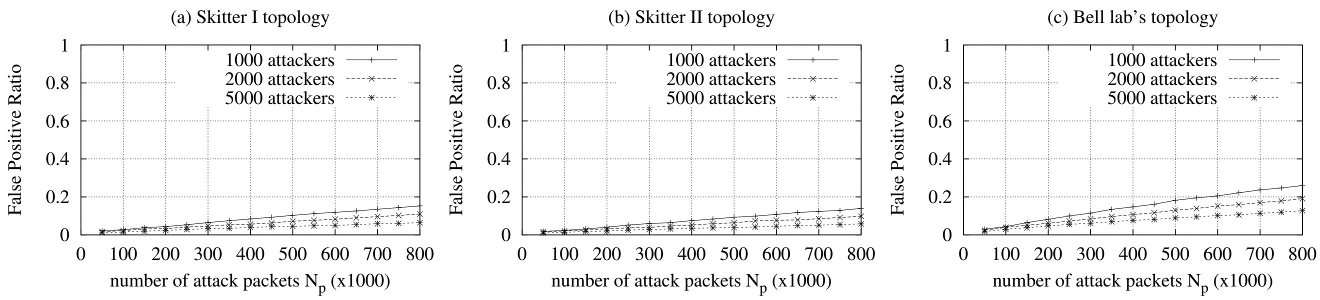


Fig. 8. False Positive Ratio of our traceback scheme on three different topologies

number of attackers and different number of attack packets used for traceback. Figures 7(a,b,c) show the FNR of our scheme against the number of attack packets $N_p$ used for traceback, under the three aforementioned Internet topologies. Similarly, Figures 8(a,b,c) show 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% ($12 \times 3.3\% = 0.4$), which correspond to the optimal parameter setting prescribed by the information-theoretic framework in Section V-C2. The three curves in each figure correspond to 1,000, 2,000 and 5,000 attackers, respectively.

For all curves in Figures 7(a,b,c), we observe that as the number of attack packets used for traceback ($N_p$) increases, the FNR value decreases sharply, which corresponds to more and more infected routers being identified. On the other hand, the FPR value in Figures 8(a,b,c) increases very slowly and is always reasonable. The increase of FPR is caused by our "one-packet decoding rule". In general, the lower FNR we get from larger $N_p$ significantly outweighs the slightly higher FPR.

We also observe that our scheme can achieve very high traceback accuracy with a reasonable number of attack packets. For example in Figure 7(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 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

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

```
for each packet w
  if (w.mark1 = 1) then
    write 0 into w.mark1;
    store the digest of w, subject to a cap of p/2;
    with probability 0.5, write 1 into w.mark2;
  else if (w.mark2 = 1) then
    write 0 into w.mark2;
    write 1 into w.mark1;
    store the digest of w, subject to a cap of p/4;
  else
    with probability p/(4-3p)
      store the digest of w;
      write 1 into w.mark1;
//p1 is the fraction of packets with
// the first marking bit set to 1
if (p1 is not p/2) then
  tune it to p/2;
//p2 is the fraction of packets with only
// the second marking bit set to 1
if (p2 is not p/4) then
  tune it to p/4;
 /* make the process "stationary" */
```

Fig. 9. Two-bits random marking and sampling scheme

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.

## VII. EXTENSION: TWO-BITS RANDOM MARKING AND SAMPLING SCHEME

We have shown that our ORMS scheme can achieve a high correlation factor of over 50% by using only one bit to

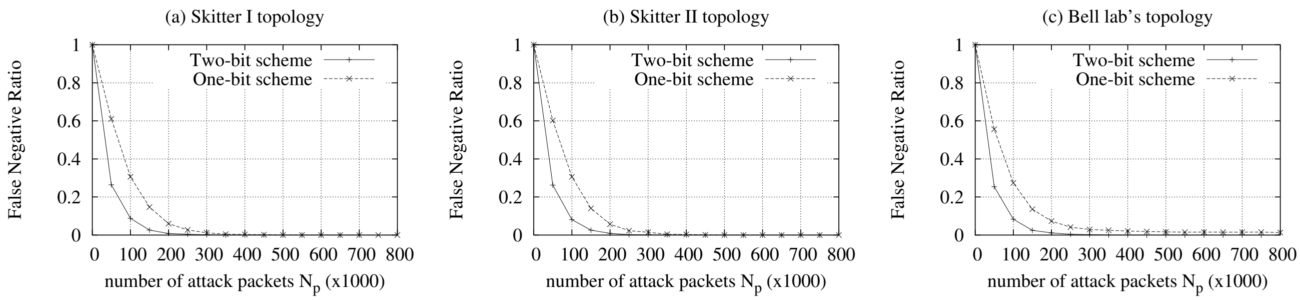


Fig. 10. False Negative Ratio of two traceback schemes on three different topologies

coordinate the sampling operations at two neighboring routers. An interesting question is whether we can further improve this correlation factor if more than one bit is used. In this section, we show that the answer is "yes". In particular, we present a scheme that can achieve a correlation factor over 75% using only two bits for coordination.

Our Two-bit Random Marking and Sampling (TRMS) scheme is shown in Figure 9. Let us denote the two bits used for coordination as $mark1$ and $mark2$. When a packet arrives at a router, depending on the value of $mark1$ and $mark2$, the router performs the following operations. If the bit $mark1$ is marked (i.e., equal to 1), the router will unmark $mark1$, store the packet in its Bloom filter, and with probability 0.5 mark the bit $mark2$. If $mark2$ is marked but $mark1$ is not, the router will store the packet in Bloom filter, unmark $mark2$, and mark $mark1$. If neither is marked, the router will sample and store the packet in the Bloom filter with probability $\frac{p}{(4-3p)}$, and mark $mark1$.

Similar to the ORMS scheme introduced in Section IV-A2, our TRMS scheme maintains the following two invariants: (1) the fraction of packets from a given router that have $mark1$ set to 1 is approximately $\frac{p}{2}$; (2) the packet sampling rate of a router is approximately $p$. In addition, we have a third invariant, that is, approximately $\frac{p}{4}$ fraction of the packets from a router will have $mark2$ marked. The "jump-start" mechanism for a router that is one hop away from the attacker is similar to the one explained in Section IV-A2. These three invariants make the correlation factor a little more than 75% between two neighboring routers, $R_1$ (downstream) and $R_2$ (upstream) when $R_2$ is not one hop away from the attacker. This is because $R_1$ samples and stores (i) all $\frac{p}{2}$ fraction of packets that $R_2$ has sampled and set $mark1$ to 1, (ii) all $\frac{p}{4}$ fraction of packets that $R_2$ has sampled and set $mark2$ to 1, and (iii) approximately $\frac{p}{(4-3p)}$ fraction of the remaining $\frac{p}{4}$ fraction of packets that $R_2$ has sampled but unmarked. Therefore, the total fraction of the packets that both $R_1$ and $R_2$ sample and store is $\frac{p}{2}+\frac{p}{4}+\frac{p}{4}\frac{p}{(4-3p)} = \frac{p(3-2p)}{4-3p}$. The correlation factor is therefore $\frac{p(3-2p)}{4-3p}$ (sampled by both) divided by $p$ (sampled by $R_2$), which is $\frac{3-2p}{4-3p}$. Note that this correlation factor is slightly larger than 75%, which is larger than achieved by our one-bit scheme, which is by slightly larger than 50%.

We compare TRMS with ORMS in terms of FNR using simulation. In this simulation, we assume that there are 1,000 attackers. We use the same three network topologies and the same configuration for $k$ (the number of hash functions in the Bloom filter) as used in ORMS[13]. The simulation results are reported in Figure 10. We can observe that the TRMS scheme clearly outperforms the ORMS scheme in terms of the number of packets needed to achieve a certain FPR. For example in Figure 10(a), when we use 200,000 of attack packets for traceback, the FNR of ORMS is around 0.06. But in the TRMS scheme, we can get a similar FNR by using only around 120,000 packets. This represents 40% decrease in the number of attack packets required to achieve similar level of false negatives. In terms of FPR, both schemes perform almost the same because false positives of the traceback are determined by the false positive rate of the Bloom filter, and in both schemes, the same Bloom filter is used and $p$ fraction of packets are sampled.

Note that we can further improve the correlation factor by using more than two bits for coordination. However, the cost of maintaining the (sampling) rates of each marked bit (line 17 and 21 in Figure 9) would become higher while the improvement on the correlation is marginal. Therefore, we do not recommend the use of more than 2 bits.

## VIII. 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 can be low enough for the scheme to scale to very high link speed (e.g., OC-768). To achieve high traceback accuracy despite the low sampling rate, we introduce ORMS, a novel sampling technique which makes use of only one marking bit in the IP header of a packet. It significantly increases the correlation between the packets sampled by 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 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. We further extend our sampling

[13] Our experimental results show that the same $k$ optimized for ORMS scheme using formula (5) in Section V also delivers the optimal results in TRMS scheme.

scheme to take two marking bits into account. 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%.

## APPENDIX
## COMPUTING $H(Z|X_{t_1}+X_{f_1}, Y_t+Y_f)$

The number of attack packets $X_{t_1}$ sampled by router $R_1$ is a binomial random variable with probability mass function $\Pr[X_{t_1}=k]=\binom{N_p d_1}{k}p^k(1-p)^{N_p d_1-k}$. The number of false positives $X_{f_1}$ when $L_v$ is queried against the Bloom filter at router $R_1$ is also a binomial random variable, with the following probability mass function:

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

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:

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

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 little 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, $W_1$ and $W_2$ have probability distributions $Binom(N_p d_2-X_{t_2}, p/(2-p))$ and $(N_p d_1-N_p d_2, p)$ respectively. Intuitively, the attack packets sampled by $R_1$ consist 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 neighbors other than $R_2$. We assume $d_1=d_2$ as explained in Section V-C1. 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 $\Pr[X=i,Y=j|Z=1]$, then according to formula (2) in Section V-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

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

Finally, note that $\Pr[X=i,Y=j,Z=a]=\Pr[X=i,Y=j|Z=a]\Pr[Z=a]$ for $a=0,1$, and $\Pr[Z=0]=\Pr[Z=1]=1/2$ as assumed in Sec. V-C1.

## REFERENCES

[1] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, B. Schwartz, S. T. Kent, and W. T. Strayer, "Single-packet IP traceback," *IEEE/ACM Transactions on Networking*, vol. 10, no. 6, pp. 721–734, Dec. 2002.

[2] L. Garber, "Denial-of-service attacks rip the Internet," *IEEE Computer*, vol. 33, no. 4, pp. 12–17, Apr. 2000.

[3] D. McGuire and B. Krebs, "Attack on internet called largest ever," http://www.washingtonpost.com/wp-dyn/articles/A828-2002Oct22.html, Oct. 2002.

[4] T. Doeppner, P. Klein, and A. Koyfman, "Using router stamping to identify the source of IP packets," in *Proc. ACM CCS*, Nov. 2000, pp. 184–189.

[5] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Network support for IP traceback," *IEEE/ACM Transactions on Networking*, vol. 9, no. 3, pp. 226–237, Jun. 2001.

[6] D. Song and A. Perrig, "Advanced and authenticated marking schemes for IP traceback," in *Proc. IEEE INFOCOM*, Apr. 2001, pp. 878–886.

[7] D. Dean, M. Franklin, and A. Stubblefield, "An algebraic approach to IP traceback," in *Proc. NDSS*, Feb. 2001, pp. 3–12.

[8] M. T. Goodrich, "Efficient packet marking for large-scale IP traceback," in *Proc. ACM CCS*, November 2002, pp. 117–126.

[9] B. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the Association for Computing Machinery*, vol. 13, no. 7, pp. 422–426, 1970.

[10] M. Roesch, "Snort - lightweight intrusion detection for networks," http://www.snort.org.

[11] H. Wang, D. Zhang, and K. G. Shin, "Detecting SYN flooding attacks," in *Proc. IEEE INFOCOM*, Jun. 2002, pp. 1530–1539.

[12] A. Hussain, J. Heidemann, and C. Papadopoulos, "A framework for classifying denial of service attacks," in *Proc. ACM SIGCOMM*, Aug. 2003, pp. 99–110.

[13] H. Burch and B. Cheswick, "Tracing anonymous packets to their approximate source," in *Proc. USENIX LISA*, Dec. 2000, pp. 319–327.

[14] M. Adler, "Tradeoffs in probabilistic packet marking for ip traceback," in *Proc. ACM Symposium on Theory of Computing (STOC)*, May 2002.

[15] T. M. Cover and J. A. Thomas, *Elements of information theory*. Wiley, 1991.

[16] R. Mahajan, S. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker, "Controlling high bandwidth aggregates in the network," *ACM Computer Communication Review*, vol. 32, no. 3, pp. 62–73, Jul. 2002.

[17] F. Kargl, J. Maier, S. Schlott, and M. Weber, "Protecting web servers from distributed denial of service attacks," in *Proc. 10th Intl. WWW Conference*, May 2001, pp. 514–524.

[18] D. K. Yau, J. C. Lui, and F. Liang, "Defending against distributed denial-of-service attacks with max-min fair server-centric router throttles," in *Proc. IEEE International Workshop on Quality of Service*, May 2002, pp. 35–44.

[19] 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*, vol. 52, no. 2, pp. 195–208, Feb. 2003.

[20] M. Sung and J. Xu, "IP Traceback-based Intelligent Packet Filtering: A Novel Technique for Defending Against Internet DDoS Attacks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 14, no. 9, pp. 861–872, Sep. 2003, preliminary version appeared in Proc. 10th IEEE ICNP.

[21] A. Yaar, A. Perrig, and D. Song, "Pi: A path identification mechanism to defend against DDoS attacks," in *Proc. IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2003, pp. 93–107.

[22] P. Ferguson, *Network Ingress Filtering: Defeating Denial of Service Attacks Which Employ IP Source Address Spoofing*, RFC 2267, Jan. 1998.

[23] 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*, Aug. 2001, pp. 15–26.

[24] A. D. Keromytis, V. Misra, and D. Rubenstein, "SOS: Secure overlay services," in *Proc. ACM SIGCOMM*, Aug. 2002, pp. 61–72.

[25] J. Mirkovic, G. Prier, and P. Reiher, "Attacking DDoS at the source," in *Proc. IEEE ICNP*, Nov. 2002, pp. 312–321.

[26] C. Papadopoulos, R. Lindell, J. Mehringer, A. Hussain, and R. Govidan, "COSSACK: coordinated suppression of simultaneous attacks," in *DISCEX III*, April 2003, pp. 22–24.

[27] J. Mirkovic, M. Robinson, P. Reiher, and G. Kuenning, "Alliance formation for ddos defense," in *Proc. New Security Paradigms Workshop, ACM SIGSAC*, Aug. 2003.

[28] C. Jin, H. Wang, and K. G. Shin, "Hop-count filtering: An effective defense against spoofed DDoS traffic," in *Proc. ACM CCS*, October 2003, pp. 30–41.

[29] D. Moore, G. M. Voelker, and S. Savage, "Inferring Internet Denial-of-Service activity," in *USENIX Security Symposium*, 2001, pp. 9–22.

[30] V. Paxson, "An analysis of using reflectors for distributed denial-of-service attacks," *ACM Computer Communications Review (CCR)*, vol. 31, no. 3, pp. 38–47, Jul. 2001.

[31] N. Duffield and M. Grossglauser, "Trajectory sampling for direct traffic observation," *IEEE/ACM Transactions on Networking*, vol. 9, no. 3, pp. 280–292, 2000.

[32] J. Turner, "New directions in communications (or which way to the information age?)," *IEEE Communications Magazine*, vol. 25, no. 10, pp. 8–15, Oct. 1986.

[33] L. Fan, P. Cao, J. Almeida, and A. Broder, "Summary cache: A scalable wide-area Web cache sharing protocol," *IEEE/ACM Transactions on Networking*, vol. 8, no. 3, pp. 281–293, 2000.

[34] S. Cohen and Y. Matias, "Spectral bloom filters," in *Proc. ACM SIGMOD Conference on Management of Data*, 2003, pp. 241–252.

[35] "CAIDA's Skitter project web page," Available at http://www.caida.org/tools/measurement/skitter/.

[36] B. Cheswick, "Internet mapping," Available at http://cm.bell-labs.com/who/ches/map/dbs/index.html, 1999.


PLACE PHOTO HERE

**Minho Sung** received a B.S. degree in computer science from Inha University, Inchon, Korea, in 1995. He received M.S. and Ph.D. degrees in computer science from Georgia Institute of Technology in 2001 and 2006, respectively. In 2006, he joined Telecommunication R&D Center, Samsung Electronics Company, Ltd. Suwon, Korea. His research interests include security and design of protocols in high-speed network.

PLACE PHOTO HERE

**Jun (Jim) Xu** is an Associate Professor in the College of Computing at Georgia Institute of Technology. He received his Ph.D. in Computer and Information Science from The Ohio State University in 2000. His current research interests include data streaming algorithms for the measurement and monitoring of computer networks, algorithms and data structures for computer networks, network security, and performance modeling and simulation. He received the NSF CAREER award in 2003 for his ongoing efforts in establishing fundamental lower bound and tradeoff results in networking. He is a co-author of a paper that won the Best Student Paper Award from 2004 ACM Sigmetrics/IFIP Performance joint conference, and the faculty advisor of the student winners. He received 2006 IBM faculty award for making fundamental contributions to performance evaluation methodologies.

PLACE PHOTO HERE

**Jun Li** received his Master of Science in Computer Science from Georgia Institute of Technology in 2002.

PLACE PHOTO HERE

**Li (Erran) Li** received a B.E. in Automatic Control from Beijing Polytechnic University in 1993, a M.E. in Pattern Recognition from the Institute of Automation, Chinese Academy of Sciences, in 1996, and a Ph.D. in Computer Science from Cornell University in 2001 where Joseph Y. Halpern was his advisor. During his graduate study at Cornell University, he worked at Microsoft Research and Bell Labs Lucent as an intern, and at AT&T Research Center at ICSI Berkeley as a visiting student. He is presently a member of the Networking Research Center in Bell Labs. His research interests are in networking with a focus on wireless networking and mobile computing. He has served as a program committee member for several conferences including ACM MobiCom, ACM MobiHoc, IEEE INFOCOM and IEEE ICNP. He is an editor of WINET and a guest editor for JSAC special issue on Non-Cooperative Behavior in Networking. He has published over 30 papers.