A More Practical Approach for Single-Packet IP Traceback using Packet Logging and Marking

Chao Gong, *Student Member*, IEEE and Kamil Sarac, *Member*, IEEE

***Abstract*— Tracing IP packets back to their origins is an important step in defending the Internet against denial-of-service (DoS) attacks. Two kinds of IP traceback techniques have been proposed as packet marking and packet logging approaches. In packet marking, routers probabilistically write their identification information into the forwarded packets. This approach incurs little overhead but requires a large flow of packets to collect the complete path information. In packet logging, routers record the digests of the forwarded packets. This approach makes it possible to trace even a single packet and, hence, is considered more powerful. At routers forwarding a large volume of traffic, however, the high storage overhead and access time requirement for recording packet digests introduce practicality problems.**

In this paper, we present a novel scheme to improve the practicality of log-based IP traceback by reducing its overhead on routers. Our approach makes an intelligent use of packet marking to help improve the scalability of log-based IP traceback. We use mathematical analysis and simulations to evaluate our approach. Our evaluation results show that, compared to the state-of-the-art log-based approach called Source Path Isolation Engine (SPIE), our approach maintains the ability to trace a single IP packet while reducing the storage overhead by half and the access time overhead by a factor of the number of neighboring routers.

***Index Terms*— Internet security, denial-of-service (DoS) attack, IP traceback, packet logging, packet marking.**

I. Introduction

Denial-of-service (DoS) attacks have been threatening the utility of the Internet severely [1]. In 2002, a coordinated attack on the Internet name service infrastructure showed the possibility and potential impact of such dedicated attacks [2]. More recently, it was reported that DoS attacks have been used as a means of extortion and become the subject of lawsuits [3]. Defending against DoS attacks requires not only measures for mitigating the effects of the attacks but also mechanisms for identifying the entities accountable for such attacks.

DoS attacks can be classified into *flooding attacks* and *software exploits* [4]. Flooding attacks work by flooding a victim with large amounts of packets while software exploits attack a victim by sending as few as a single packet. Note that software exploits cover a wide spectrum of attacks where the attacker can use weaknesses of a service running on a victim machine or can exploit vulnerabilities emerging from the semantics of a networking protocol, e.g., TCP's vulnerability to connection reset attacks. In addition, an attacker can place an arbitrary IP address into the source address field of an IP packet (known as *IP spoofing*) to hide the attack origin. IP spoofing makes it harder to defend against DoS attacks.

Tracing the paths of IP packets back to their origin, known as *IP traceback*, is an important step in defending against DoS attacks employing IP spoofing. IP traceback facilitates holding attackers accountable and improving the efficacy of mitigation measures. Most of the recently publicized attack incidents have been flooding based DoS attacks. Accordingly, most of the work in IP traceback has focused on building efficient approaches to trace back flooding based DoS attacks (see Section II-D). On the other hand, an IP packet is the smallest unit of communication in the Internet. The existence of protocols/services that are vulnerable to packet injection attacks necessitates an ability to trace a single packet back to its origin. Therefore, it is desirable for an IP traceback approach to be able to trace the path of a single IP packet. Single packet traceability helps in identifying the origin of both flooding and software exploit based DoS attacks.

The existing approaches for IP traceback can be grouped in two orthogonal dimensions: packet marking [5] and packet logging [6]. The main idea behind packet marking is to record network path information in packets. In mark-based IP traceback, routers write their identification information (e.g., IP addresses) into a header field (hereinafter termed "marking field") of forwarded packets. The destination node then retrieves the marking information from the received packets and determines the network path. Due to the limited space of the marking field, routers probabilistically decide to mark packets so that each marked packet carries only a partial path information. The network path can be constructed by combining the marking information collected from a number of received packets. This approach is also known as probabilistic packet marking (PPM) [7]. PPM incurs little overhead at routers. However, it requires a flow of marked packets to construct the network path toward their origin.

The basic idea in packet logging is to record the path information at routers. In log-based IP traceback, packets are logged by the routers on the path toward the destination. The network path is then derived based on the logged information at the routers. Compared to mark-based IP traceback, the log-based approach is more powerful as it can trace attacks that use a single packet, i.e., software exploit attacks, along with flooding attacks [8]. Historically, packet logging was thought impractical due to the enormous storage space required for packet logs. Snoeren *et al.* [9] proposed a hash-based IP traceback approach, called Source Path Isolation Engine (SPIE), to realize log-based IP traceback in practice. Their approach reduces the storage overhead significantly

Chao Gong is with the Dept. of Computer Science at Univ. Mary Hardin-Baylor and Kamil Sarac is with the Dept. of Computer Science at Univ. Texas Dallas (emails: cgong@umhb.edu and ksarac@utdallas.edu) (this work was done when Dr. Gong was a Ph.D. student at UT Dallas)

through recording packet digests in a space-efficient data structure, a Bloom filter [10]. SPIE has made a significant improvement on the practicality of log-based IP traceback. However, its deployment at high speed networks has still been a challenging task due to the high storage overhead and access time requirement for recording packet digests. Considering the effectiveness of log-based IP traceback in tracing both flooding and software exploit attacks, there is a need to develop more scalable solutions to facilitate its deployment at high speed networks.

In this paper, we present a novel hybrid IP traceback approach based on both packet logging and packet marking. Our main design goal is to maintain the single-packet traceback ability of hash-based approach and, at the same time, alleviate the storage overhead and access time requirement for recording packet digests at routers. From this perspective, the main contribution of our work is to improve the practicability of single-packet IP traceback by decreasing its overhead. This paper builds on our previous work published as a conference paper [11]. In the current version, we extend the basic idea to design a complete system and provide a detailed discussion on various architectural and operational components. We also significantly extend the scope of the performance analysis and conduct simulations to back up our analytic results.

The key idea of our approach is to record network path information partially at routers and partially in packets. While a packet is traversing the network, the most recent portion of the network path is recorded in the packet, and the upstream portion of the path is recorded at some intermediate routers. In our approach, depending on the availability of free space in the marking field of the forwarded packets, routers decide where to record network path information. If there is free space available in the marking field, routers write their identification information into the packets; otherwise, routers compute and record the packet digests (the path information stored in the marking field is also encoded into the digests), and then clear the marking field. Based on this idea, we develop a concrete single-packet IP traceback approach. Compared to SPIE, our approach (1) reduces the storage overhead of packet digests to one half and (2) reduces the access time requirement for recording packet digests by a factor of $2n$, where n stands for the number of neighboring routers.

We evaluate our approach by comparing it to the existing single-packet traceback and hybrid traceback approaches. For the former comparisons, we use both analysis and experimentation. For the analytical part, we develop a mathematical model to show the packet logging overhead and traceback process overhead. We also study the traceback accuracy by mathematically formulating the false positive rate and discuss potential ways to compensate for inaccuracies. For the experimental part, we use simulations to supplement our analysis in measuring the storage, access time, and traceback overhead. For the hybrid IP traceback comparisons, we use analysis to compare our approach with two recently proposed hybrid traceback approaches [12] based on their capabilities and overhead. Our evaluation results indicate the superiority of our approach over the existing approaches in both categories.

The rest of this paper is organized as follows. Section II presents the background information and related work. Section III describes our IP traceback approach in detail. Section IV evaluates the performance of our approach through mathematical analysis and simulation. Section V discusses deployment issues. Finally, Section VI summarizes the paper.

II. Background and Related Work

A. Background

In this paper, we refer to a router with high speed links as a *high-speed router*. We also term a packet of interest an *attack packet*. Similarly, the source and destination of an attack packet is an *attacker* and a *victim*, respectively. The sequence of routers traversed by an attack packet on its way from source to destination make up an *attack path*. The attack path from the attacker to the victim is represented as an ordered list of routers $(R_1, R_2, \ldots, R_m)$. The objective of IP traceback is to figure out this ordered list of routers. The process of constructing attack paths is called *traceback process*.

Based on the vulnerability that is exploited, DoS attacks can be classified into *flooding attacks* and *software exploits* [4]. Flooding attacks (e.g., smurf, SYN flood) work by flooding victims with large amounts of traffic. Flooding attacks consume some limited resource (e.g., link bandwidth or computing resource) at victims to prevent legitimate users from accessing that resource. Software exploits (e.g., teardrop, ping-of-death) work by sending victims a single or a few malformed packets to abuse some feature or implementation bug of operating systems or applications to disable the service.

Reckoning with the current Internet environment, we prefer an IP traceback approach with the following features:

1) Ability to trace a single packet. This very feature enables the IP traceback approach to trace both flooding and software-exploit DoS attacks.
2) Robustness against attacks. Attackers may be aware of and try to compromise the IP traceback approach. Robustness against such attacks is desirable.
3) Backward compatibility. IP packets may undergo valid transformation (e.g., fragmentation, tunneling) while traversing the network. The IP traceback approach should operate in the presence of such transformations.
4) Financial motivation. Internet Service Providers (ISP) prefer value-added services which can create new revenue streams. The IP traceback approach should be suitable to be deployed as a revenue-generating service.
5) Low overhead on routers. The overhead imposed on the deploying routers should be acceptable.

B. Hash-based IP Traceback (SPIE)

In SPIE, routers compute and store digest for each forwarded packet. Packet digests are stored into a time-stamped digest table which is implemented with a space-efficient data structure, a Bloom filter [10]. When becoming saturated, the digest table is swapped out for an empty table and is archived for some period of time for potential traceback process. [1].

[1]Depending on the implementation, the archived digest tables may be stored at local routers [9] or transferred to remote traceback servers [13].

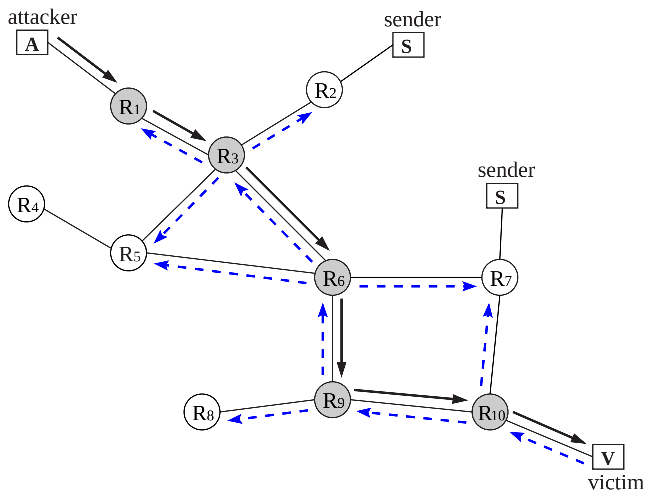

Fig. 1. Traceback process in SPIE. Solid arrows represent the attack path; dashed arrows represent traceback queries.

Each digest table is annotated with the time interval which the table covers and the hash functions used for computing packet digests during that interval.

A traceback server (or multiple servers in hierarchy) equipped with network topology information is responsible for conducting the traceback process. The traceback server constructs the attack path by querying routers hop-by-hop, starting from the last-hop router (the router next to the victim) toward the source. [2]. In each step, based on the responses from the routers, the traceback server identifies one router on the attack path and decides the neighboring routers to be queried in the next step. The traceback server queries routers by sending query messages including the attack packet and the time when the attack packet arrived at the downstream router on the attack path (for the last-hop router, it is the time when the victim received the attack packet). The router being queried computes the packet digest and examines local digest tables of the relevant time period. If the packet digest is found in a digest table, the router is considered to be on the attack path, and the packet arriving time is estimated as the latest possible time in the time interval covered by the digest table. Fig. 1 illustrates the hash-based IP traceback process.

SPIE has the following advantages. First, it can determine the network path of a single packet. Second, it is robust against attacks. In SPIE, the network path information is stored at routers in the form of packet digests. It is not easy to compromise routers to garble packet digests or to induce routers to produce specific digests. Third, SPIE does not interfere with the current version of the IP protocol and can trace packets undergoing transformation. Last, SPIE appeals to ISPs. In SPIE, the traceback process is requested by end hosts and accomplished by ISPs who can offer IP traceback as a revenue-generating service.

The increasing amount of traffic load at the backbone networks introduces practicality problems for SPIE. First, a huge amount of memory is required to store packet digests at high-speed routers. The high storage requirement limits the time duration for which packet digests can be kept at routers and therefore the window of time in which attack packets can be traced successfully. Second, packet digests must be recorded into digest tables at a rate commensurate with packet arriving rate. In practice, the access time requirement places limits on the memory type of digest tables. High-speed routers require SRAM digest tables which are 8 to 16 times as expensive as DRAM. The current technology limits the size of SRAM digest table to be smaller than that of DRAM table. Hence a SRAM digest table records a shorter time period of traffic. This implies that high-speed routers may need to examine more digest tables to check whether it has forwarded an attack packet within an indicated time period. The more digest tables examined, the greater the possibility to get false positive results.

C. Probabilistic Packet Marking (PPM)

Compared to SPIE, the PPM approach has been studied widely [7], [14], [15], [16], [17], [18]. In PPM, a router marks packets with its identification information as they pass through that router. The marking value overloads a rarely used field in IP header, i.e., 16-bit IP identification field. Since the marking field is too small to record the entire path, routers mark packets with a probability so that each marked packet carries a partial path information. The whole network path can be constructed by combining a number of such packets.

PPM does not incur any storage overhead at routers and the marking procedure (a write and checksum update) can be easily executed at current routers. However, due to its probabilistic nature, the PPM approach inherently needs multiple packets originated from an attacker to construct the attack path. For example, the current state-of-the-art PPM approach, FIT, requires tens of packets to identify an attack path with high probability [18]. This feature renders PPM approaches suitable for tracing flooding DoS attacks only. Furthermore, due to probabilistic marking, attack packets may arrive at victims without having been marked by any of the intermediate routers. Wily attackers can confuse the victim by sending packets with carefully forged marking values to mislead the traceback process. Because the IP identification field designated for IP fragmentation is reused for marking information, the PPM approach collides with fragmented IP traffic. When a packet undergoes a transformation (e.g., tunneling), the marking information in the packet header will be lost. Finally, in PPM approaches, it is the end systems that use the collected marks to build an attack path. Hence, it is difficult for ISPs to come up with a business model to sell PPM-based IP traceback as a value-added service to their customers [19].

D. Related Work

Most existing work on IP traceback is in the direction of PPM [7],[14]-[18]. These studies can be characterized as improvements on the scalability and efficiency of mark-based IP traceback. On the other hand, the amount of the literature on single-packet IP traceback has surprisingly been limited.

[2]In the case that digest tables are archived at traceback servers, the traceback server examines the digest tables from different routers in the order as if the tables were archived at routers.

From this perspective, the work presented in this paper makes an important contribution to improve the state of the art in single-packet IP traceback.

Lee *et al.* [20] presented an approach to reduce the overhead of the hash-based approach. They proposed to digest packet aggregation units (packet flows or source-destination sets) instead of individual packets. Recording the digests of packet aggregation units reduces the digest table storage overhead. However, tracing an individual packet is accomplished by tracing the packet aggregation to which the packet belongs. Moreover, depending on the implementation, either the writing or reading rate of digest table should be commensurate with packet arriving rate. Thus, this approach does not alleviate the access time requirement.

Li *et al.* [21] proposed probabilistic packet logging where routers probabilistically select a small percentage of forwarded packets to record their digests. This method reduces both the storage overhead and access time requirement for recording packet digests at routers. But the tradeoff is the loss of the ability to trace individual packets since the probability that all routers on an attack path record a specific packet is tiny.

Recently, Al-Duwairi *et al.* [12] proposed two hybrid IP traceback approaches, *distributed link-list traceback* (DLLT) and *probabilistic pipelined packet marking* (PPPM). The main design goal in these approaches is to reduce the number of packets required for constructing attack paths in the PPM approach through utilizing packet logging. In comparison, our work is to make use of packet marking to reduce the overhead of log-based IP traceback in tracing a single packet.

The basic idea in both DLLT and PPPM approaches is to preserve the marking information carried by the packet before marking a packet. When a router probabilistically decides to mark a packet, the router records the marking value carried by the packet before writing a new value into the packet. In DLLT, the preserved marking information is stored at routers and victims query routers during the traceback process. In PPPM, routers transfer those marking information to the original destinations via writing them into other packets to the same destinations. Given the common hybrid feature in both our work and the work in [12], we compare these two approaches with our approach in detail in Section IV-B.

III. Hybrid Single-Packet IP Traceback

In this paper, we propose a hybrid single-packet IP traceback approach based on both packet marking and logging. Our approach remains the same single-packet traceback capability as SPIE, but incurs less overhead at routers through utilizing packet marking. Our approach has a similar architecture as SPIE. In this architecture, routers create audit trails on network traffic, and traceback servers construct attack paths through examining those audit trails. The differences are at audit trails as well as the approach to creating and examining audit trails.

A. Main Idea

In hybrid single-packet IP traceback, each traceback-enabled router could commit both packet marking and packet logging operations. The marking operation on a packet is to append router identification information into the marking field of the packet. The logging operation on a packet is to compute and record the packet digest. When forwarding a packet, routers decide to mark or log the packet depending on whether there is free space available in the marking field of the packet. If so, routers mark the packet; otherwise, routers log the packet and clear the marking field.

Suppose that the identification information of k routers can fit into the marking field of one packet. When a router logs a packet, the marking value carried by the packet, which indicates k routers that the packet has traversed most recently, is also encoded into the packet digest. In this way, logging a packet at a router records not only the current router but also the k upstream routers on the network path. While a packet is traversing the network, logging the packet at every $(k+1)$th router is enough to record the complete network path. During the traceback process, the attack path is constructed one portion by one portion. The marking value of the attack packet received by a victim indicates the most recent portion ($\leq k$ routers) of the attack path. And querying the router having logged the attack packet identifies an upstream portion ($k+1$ routers) of the path. We depict the main idea of hybrid single-packet IP traceback with an example in Fig. 2.

In hybrid single-packet IP traceback, packets are logged at every $(k+1)$th router enroute from source to destination. Reckoning with the network traffic, $1/(k+1)$ of packets forwarded by a router, on average, need to be logged at that router. Compared to SPIE, the storage overhead and access time requirement for recording packet digests are decreased by a factor of $k+1$. In addition, the packets arriving at a router can be categorized based on their marking values and recorded into different digest tables simultaneously. That reduces the access time requirement further.

Based on the above-mentioned idea and current Internet environment, we consider a basic case of hybrid single-packet IP traceback and develop a concrete IP traceback approach termed as Hybrid IP Traceback (HIT). In HIT, the marking field of packet accommodates the identification information of a single router. While a packet is traversing the network, the routers on the path mark the packet deterministically but log the packet alternately. Note that a more carefully designed marking approach could be used to increase the inter-logging distance between routers (i.e., have more than one consecutive routers to mark a packet). However, in this paper, we only consider the simplest case to serve as a proof-of-concept. In the following, we present HIT from the below aspects:

1) Router operation: how to mark packets with the complete identification information of a router and how to compute packet digests?
2) Digest table: how to categorize packet digests so as to record them into different digest tables simultaneously?
3) Traceback process: how to decrypt the marking information encoded into packet digests?
4) Compatibility and transformation: how to achieve backward compatibility and how to trace packets undergoing transformation?

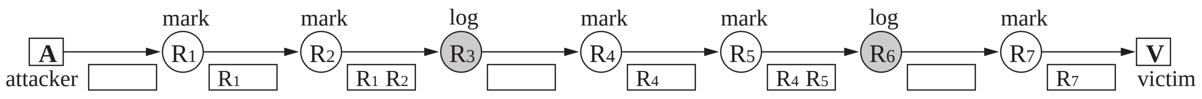

Fig. 2. Hybrid single-packet IP traceback. In this example, the marking field of packet can accommodate the identification information of two routers. Routers R_3 and R_6 log the packet, the other routers mark the packet.

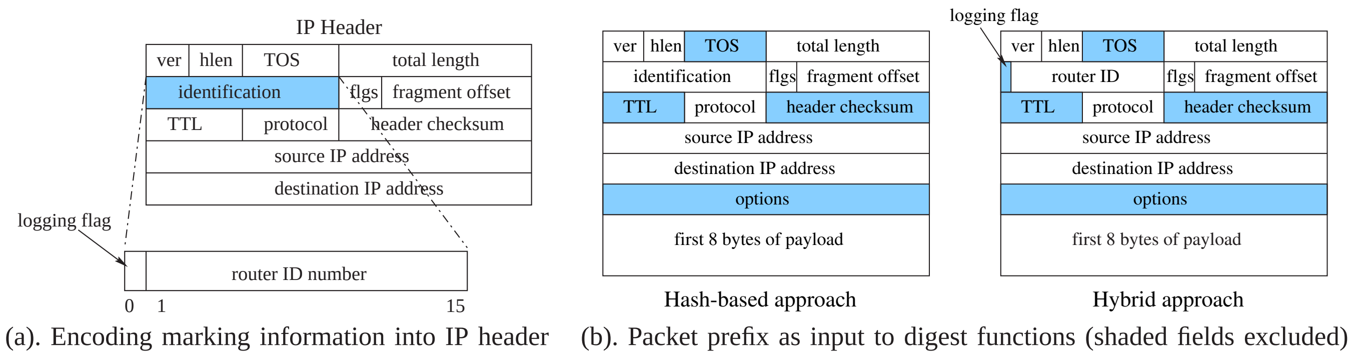

Fig. 3. Marking and logging operations on IP packets.

B. Router Operation

In HIT, each traceback-enabled router is assigned a 15-bit ID number. ID numbers are used to differentiate neighboring routers of a router. Thus, the same ID number can be assigned to any two routers as long as they do not have a common neighbor. Muthuprasanna *et al.* [22] study unique ID number assignment problem for Internet routers. Based on an analysis on several Internet topology measurement data sets, they report that a 12-bit ID field is sufficient for unique ID number assignment within two-hop neighborhood. Therefore, 15-bit IDs are more than enough for assigning a unique ID number to all neighboring routers of each router within a network. Like most PPM approaches, the marking values are encoded in the 16-bit identification field of IP header. The leftmost bit is termed *logging flag bit*. It is set to 1 if the current router commits logging operation on the packet, otherwise set to 0. The remaining 15 bits are used to store a router ID number. Fig. 3-a depicts the encoding scheme.

HIT computes packet digests in a similar way as in SPIE. A router uses an appropriate-length prefix of IP packet as input to digest functions. In SPIE, the packet prefix is the 20-byte IP header excluding 3 variant fields (TOS, TTL, and checksum) plus the first 8 bytes of payload. HIT uses almost the same packet prefix except that the logging flag bit is also excluded. Fig. 3-b shows the packet prefixes used in two approaches.

In this encoding, the ID number of upstream routers is encoded into packet digests. If a packet is logged at a router, the fact that the packet digest is stored at the router indicates that the router is on the network path. Moreover, the packet digest includes the information of the upstream router on the network path. Therefore, the logging operation on a packet records the current and the upstream routers on the path.

When forwarding a packet p, the router decides its operation on p depending on the value of p's logging flag bit. If the logging flag is 0 (the upstream router did not log p), the router chooses to commit both logging and marking operation; if the logging flag is 1 (the upstream router logged p), the router chooses to commit only marking operation.

When a packet transfers from a sender host to the first router on its network path, the logging flag is unset and therefore meaningless. The first router on the path needs additional mechanisms to decide its operation. We propose two improvements to the above-mentioned algorithm. The first one is optional, the second one is compulsory:

1) If an input port of a router is connected with only end hosts, we upgrade the input port to a *marking input port*. The marking input port is assigned an ID number as if the port were a neighbor of its router. When a packet arrives at the marking input port, the port marks the packet with its ID number and resets the logging flag.
2) Each router maintains a *neighbor list*. This list includes the ID numbers of its neighboring routers and marking input ports. For a given router R, if the router ID number i carried by a packet equals to an entry of the neighbor list at R, then i is regarded as *valid* at R.

When forwarding a packet p, the router first checks whether the router ID number i carried by p is valid. If i is valid, the router makes decision based on the logging flag in p. If i is not valid, that means p comes directly from the sender host or an attacker which sends packets with forged marking values. In this case, the router chooses to commit only marking operation. Fig. 4 describes the full algorithm.

While a packet traverses the network, the routers on the path mark the packet deterministically but log it alternately. Depending on the setup of the router and the arriving port, the packet may or may not be logged at the first router on the path. The reason of deterministic marking is to facilitate the traceback process (see Section III-D).

C. Digest Table

Similar to SPIE, HIT stores packet digests in digest tables which are implemented with Bloom filters. However, in the HIT approach, routers may maintain multiple digest tables to record multiple packet digests at the same time. Each digest table is associated with one or more router ID numbers. The

```
let $d$ be the ID number of the current router
FOR each packet $p$
    IF $p$.router_ID_number is valid at the current router THEN
        IF $p$.logging_flag = 0 THEN
            compute and store the digest of $p$
            $p$.router_ID_number := $d$
            $p$.logging_flag := 1
        ELSE
            $p$.router_ID_number := $d$
            $p$.logging_flag := 0
    ELSE
        $p$.router_ID_number := $d$
        $p$.logging_flag := 0
```

Fig. 4. Logging and marking procedure at routers.

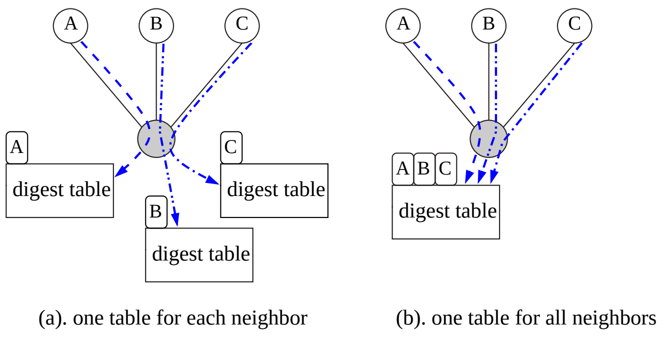
A
B
C
A
digest table
C
digest table
B
digest table
(a). one table for each neighbor
A
B
C
A
B
C
digest table
(b). one table for all neighbors

Fig. 5. Organization of digest tables. Consider a router with three neighboring routers, A, B, and C. The router may maintain one digest table for each neighboring router. If the router is a low-speed router, it has another option to maintain one digest table for all neighboring routers.

digest of a packet is stored into a digest table only if the digest table is associated with the router ID number carried by the packet.

In particular, a router may maintain a different digest table for each of its neighboring routers. That is, each digest table is associated with the ID number of one neighboring router. When the router decides to log a packet, it examines the router ID number carried by the packet, then stores the packet digest into the corresponding digest table. In this way, packets coming from different neighboring routers can be digested and recorded into different digest tables simultaneously as long as each table has its own read/write hardware. Hence, the digest table access time is not required to be commensurate with the overall packet arriving rate, but the maximum packet arriving rate from different neighboring routers.

At a low-speed router where the overall packet arriving rate is not higher than the cycle time of DRAM, the access time of digest table is not a concern. The low-speed router has another option to maintain one DRAM digest table which is associated with the ID numbers of all its neighboring routers. Packets coming from all the neighboring routers are digested and recorded into the same digest table. Fig. 5 illustrates the two extreme cases of the organization of digest tables.

After storing a certain number of packet digests which is dependent on its configuration, a digest table is regarded as *saturated*. When becoming saturated, digest tables are paged out and archived for some period of time. The length of the

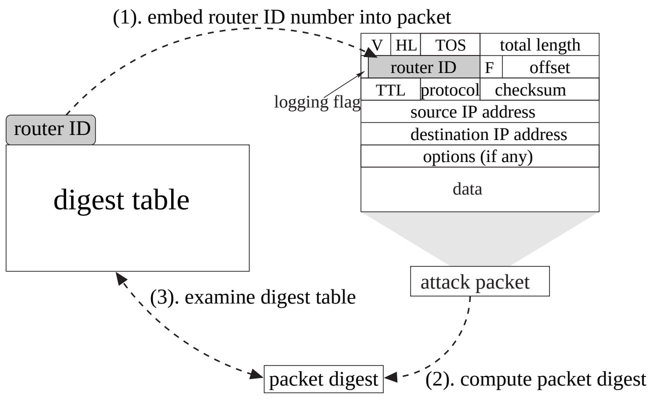

Fig. 6. Checking whether an attack packet is recorded into a digest table.

time period depends on the resource constraints of routers and the requirements of the IP traceback scheme. In addition, each digest table is annotated with its associated router ID numbers. The storage space for these router ID numbers is negligible.

D. Traceback Process

Similar to SPIE, the traceback process in HIT is managed by traceback servers equipped with network topology information. However, HIT utilizes the marking information stored in packets and digested at routers to facilitate the traceback process.

The victim under DoS attack dispatches a traceback request to the traceback server, providing an attack packet and the time of receiving the attack packet. Given a victim and an attack packet, the traceback server can pinpoint the last hop router based on the location of the victim and the router ID number carried by the packet. From the value of the logging flag bit in the packet, the traceback server can further determine whether the last hop router logged the packet. If the last hop router logged the packet, the traceback server queries that router; otherwise, the traceback server dispatches queries to the neighboring routers of the last hop router.

When a router receives a query from the traceback server, the router examines all digest tables of the relevant time period for the attack packet. In order to check whether an attack packet is recorded into a digest table, the router embeds the router ID number associated with the digest table into the packet, computes the packet digest, and consults the digest table. If an entry exists for the packet, the current router is believed to be on the attack path, and the router indicated by the router ID number is considered as the upstream router on the attack path. Fig. 6 illustrates the process of checking whether an attack packet is recorded into a digest table.

If a router commits logging operation on an attack packet, querying that router will find out two routers (i.e., this router and its upstream router) on the attack path. Upon receipt of responses from queried routers, the traceback server determines two routers on the attack path, and then dispatches queries to the neighboring routers of the furthest router having been identified. Fig. 7 illustrates the traceback process in HIT.

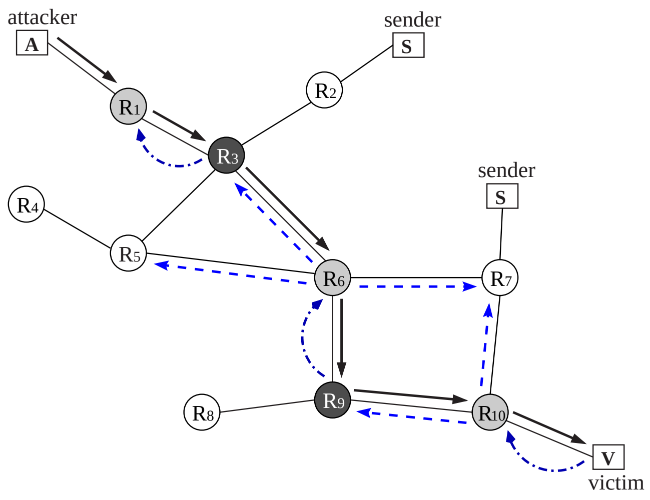

Fig. 7. Traceback process in HIT. Solid arrows represent the attack path; dashed arrows represent traceback queries; and dashed curved arrows represent decisions made from marking information. Among the routers on the attack path, R_3 and R_9 logged the attack packet; the others did not.

E. Compatibility and Transformation

With some improvements, HIT is able to achieve backward compatibility and trace packets undergoing transformation. The main idea of the improvements is that (1) routers do not mark but log IP fragments, and (2) routers both mark and log packets undergoing transformation.

Besides the ordinary digest tables described in Section III-C, each router maintains a special digest table, called *fragmentation and transformation digest table*, or FTDT. FTDT is only for storing the digests of IP fragments and the digests of IP packets having been transformed at the current router. Packet digests stored in FTDT are computed in the same manner as in SPIE. The approach to managing and examining FTDT is also the same as SPIE.

Each router also maintains a *transform lookup table*, or TLT, corresponding to FTDT. TLT records packet transformation information and is indexed by packet digests. The implementation of TLT is presented in [9]. Given a packet, consulting TLT can find out whether the packet was transformed, and if so, the original packet (or the original packet prefix used for computing packet digest) can be reconstructed.

The router processes each forwarded packet p as follows:

1) If p is an IP fragment and transformed at the current router, the router records transformation information into TLT and stores the digest of p into FTDT.
2) If p is an IP fragment and not transformed at the current router, the digest of p is stored into FTDT.
3) If p is a non-fragmented packet and transformed at the current router, the router records transformation information into TLT, stores the digest of p into FTDT, marks p with its ID number, and sets the logging flag of p to 1.
4) Otherwise, the router follows the algorithm in Fig. 4.

After receiving a query about an attack packet p, the router examines digest tables as follows:

1) If p is an IP fragment, the router examine FTDTs of the relevant time period.
2) Otherwise, the router examine all digest tables (including FTDTs) of the relevant time period.

At the same time, the router also consults TLTs for the same time intervals. If p was transformed at the router, the router inverts p to its original form $p\prime$. If $p\prime$ is a non-fragmented packet, the router can further determine its upstream router and whether the upstream router logged $p\prime$, from the marking information in $p\prime$.

The traceback server traces an attack packet p as follows:

1) If p is an IP fragment, the traceback process is the same as in SPIE. That is, the traceback server queries routers in the hop-by-hop manner.
2) If p is a non-fragmented packet, the traceback process is similar to the one presented in Section III-D. The only difference is on querying routers where packet p underwent transformation. To illustrate, if p undergoes transformation at a router R_j, R_j will log p, no matter whether its upstream router, say R_i, has logged $p\prime$, the original form of p. During the traceback process, based on the response from R_j, the traceback server can find out the original packet $p\prime$, the upstream router R_i, and whether R_i logged $p\prime$, then takes proper action as follows:
   a) If $p\prime$ is an IP fragment, go to step 1.
   b) Otherwise, if R_i logged $p\prime$, query R_i.
   c) Otherwise, query the neighboring routers of R_i.

HIT uses the same amount of resources as SPIE in recording the digests of IP fragments and the packet transformation information.

IV. Performance Evaluations

We evaluate HIT by comparing it to the state-of-the-art approaches in (1) single-packet and (2) hybrid IP traceback.

A. Comparison to Single-Packet IP Traceback Approach

We use mathematical analysis and simulation to compare HIT with the original single-packet IP traceback approach, i.e., SPIE [9]. The performance metrics include:

1) Packet logging overhead:
   - Digest table storage (*DTS*): the memory used to store packet digests at a router for a period of time.
   - Digest table access time requirement (*DTA*): the number of packet digests written into a digest table per unit time.
2) Traceback process overhead:
   - The number of queried routers (*NR*): the number of routers queried by the traceback server during the traceback process.
   - The number of examined digest tables (*ND*): the number of digest tables examined at a queried router during the traceback process.
3) Traceback accuracy:
   - The number of false positive branches (*NF*): the number of spurious branches grafted onto the attack path during the traceback process.

TABLE I
PERCENTAGES OF DIFFERENT TYPES OF IP PACKETS.

Packet Type	Percentage
1. IP fragments	α
2. non-fragmented packets to be logged at this router (includes 2(a) & 2(b) below)	$(1-\alpha)Y$
2(a). non-fragmented packets not logged at the upstream routers	$(1-\alpha)(1-Y)$
2(b). non-fragmented packets logged at the upstream routers and transformed at this router	$(1-\alpha)Y\beta$

From the perspective of ISPs, the packet logging overhead is more crucial than the traceback process overhead and traceback accuracy. Routers keep recording packet digests no matter whether DoS attacks are reported or not. In general, the traceback process is not a frequent operation and ISPs can charge for traceback processes. Additional mechanisms may be employed to refine traceback results.

1) Packet Logging Overhead: In HIT, packets logged at a router include

1) IP fragments,
2) non-fragmented packets which need to be logged at the router, which include:
   a) non-fragmented packets which have not been logged at the upstream routers,
   b) non-fragmented packets logged at the upstream routers and transformed at the current router.

Let us consider all packets forwarded by a router. We use P_l to denote the percentage of packets which need to be logged at the router. We also assume the percentage of IP fragments is α, and the percentage of IP packets undergoing transformation at the router is β. Furthermore, let us consider all non-fragmented packets forwarded by a router. We use Y to denote the percentage of packets which need to be logged at the router out of the non-fragmented packets. Then, the percentages of different types of IP packets in all packets forwarded by a router can be expressed as in Table I.

IP packets to be logged at a router include the packets of types 1 and 2 as listed in Table I. Thus we have

$$P_l = \alpha + (1-\alpha)Y \ . \tag{1}$$

Expressing Y in terms of P_l, we have

$$Y = \frac{P_l - \alpha}{1-\alpha} \ . \tag{2}$$

Type 2 includes 2(a) and 2(b), thus

$$P_l = \alpha + (1-\alpha)(1-Y) + (1-\alpha)Y\beta \ . \tag{3}$$

We use (2) to substitute for Y in (3). The result is

$$P_l = \frac{1+\alpha-\alpha\beta}{2-\beta} \ . \tag{4}$$

Because $0 \le \alpha \le 1$ and $0 \le \beta \le 1$, we know $0 \le \alpha\beta \le 1$ and $0 \le \alpha - \alpha\beta \le 1$. Applying these inequalities to (4) yields

$$\frac{1}{2} \le P_l \le \frac{1+\alpha}{2-\beta} \ . \tag{5}$$

Measurement studies show $\alpha \le 0.25\%$ [23] and $\beta \le 3\%$ [24]. Thus we have

$$0.50 \le P_l \le 0.51 \ . \tag{6}$$

That means, in HIT, about 50% of packets forwarded by a router need to be logged at that router.

In SPIE, all packets forwarded by a router need to be logged. Therefore, the digest table storage and access time requirement in HIT are roughly one half of those in SPIE. Let DTS_y and DTS_a denote the digest table storage in HIT and in SPIE, respectively, then

$$DTS_y = P_l \times DTS_a \cong \frac{1}{2} \times DTS_a \ . \tag{7}$$

In addition, in HIT, a router can keep separate digest table for each neighboring router and packets coming from different neighboring routers can be recorded in corresponding digest tables in parallel. Hence the digest table access rate can be reduced further by a factor of the number of neighboring routers. Suppose a router has n neighboring routers. In the best case, the traffic arrives at the router equally from each of its neighbors, then

$$DTA_y = P_l \times \frac{1}{n} \times DTA_a \cong \frac{1}{2n} \times DTA_a, \tag{8}$$

where DTA_y and DTA_a represent the digest table access time requirement in the HIT and SPIE, respectively. In the worst case, all arriving traffic is from one neighbor, then

$$DTA_y = P_l \times DTA_a \cong \frac{1}{2} \times DTA_a. \tag{9}$$

2) Traceback Process Overhead: During the traceback process, the traceback server queries routers in the network and routers being queried examine local digest tables. The number of queried routers reflects the overhead on the traceback server and the number of examined digest tables reflects the overhead on the queried routers.

For a given attack path $(R_1, R_2, \ldots, R_m)$, the traceback process is to construct the attack path backward, from R_m to R_1, through iteratively querying the neighboring routers of the furthest router having been identified. In HIT, a packet traversing the network is generally logged at every other router on the path. Suppose router R_i on the attack path logged the attack packet. Querying router R_i can identify two routers on the attack path, namely, R_i and R_{i-1}. Suppose an attack path has h hops and each router on the attack path has n neighboring routers on average. During the traceback process, the traceback server needs to dispatch $h/2$ rounds of queries, querying $n-1$ routers in each round (excluding the downstream router on the attack path), totally $(n-1) \times h/2$ routers.

In SPIE, a packet traversing the network is logged at every router on the network path and querying a router on the attack path can only identify that router itself. With the same assumption above, during the traceback process, the traceback server needs to dispatch h rounds of queries, querying $n-1$ routers in each round, totally $(n-1) \times h$ routers. Using NR_y

and NR_a to denote the number of routers queried during the traceback process in HIT and in SPIE respectively, we have

$$NR_y = \frac{1}{2} \times NR_a. \tag{10}$$

The traceback server queries routers through sending query messages. The query message includes an attack packet p and the time t when the attack packet arrived at the downstream router on the attack path. Based on time t and transmission delay between routers, the queried router is supposed to infer the time when packet p was digested at the current router and then examine relevant digest tables. However, because of timing uncertainties (e.g., varying transmission delay and time asynchronization between routers), the queried router can only estimates a *query time period* Δt which covers the time when the packet was digested. The router then examines all cached digest tables whose time intervals overlap Δt.

If the size of a digest table is s bits and its memory efficiency factor[3] is set to be r, that digest table can store $s \times r$ packet digests. Suppose the number of packet digests written into the digest table per unit time is u, then the time interval covered by the digest table can be computed as

$$t = \frac{s \times r}{u} \ . \tag{11}$$

In general, the time interval covered by a digest table in HIT is longer than that in SPIE due to the following reasons:

1) HIT stores fewer packet digests at routers. In HIT, routers need to log roughly one half forwarded packets while SPIE requires routers to log every forwarded packet.
2) HIT distributes packet digests into multiple digest tables. In HIT, the router maintains a different digest table for each neighboring router. Packets coming from different neighbors are recorded into different digest tables in parallel. In SPIE, packets from different neighbors are recorded into the same digest table.
3) It may be the case that HIT uses digest tables of larger size. SPIE requires a higher rate to write packet digests into digest tables. At some high-speed routers, it is possible that DRAM digest tables are suitable for HIT but not for SPIE. SPIE requires SRAM digest tables instead. The current technology limits the size of SRAM digest table to be smaller than that of DRAM table.

Consider a router with n neighboring routers. Let P_l be the percentage of packets to be logged at the router. We have

$$u_y = \frac{P_l}{n} \times u_a \ , \tag{12}$$

where u_y and u_a denote the average packet digest writing rate of digest tables in HIT and in SPIE, respectively. Let s_y and s_a be the digest table size in HIT and in SPIE, respectively. We assume the ratio of the digest table size in HIT to that in SPIE is c $(c \geq 1)$. Thus, we obtain

$$s_y = c \times s_a \ . \tag{13}$$

[3]See Section IV-A.3 for the definition of memory efficiency factor.

We apply (12) and (13) to (11). Then the relationship between the average time intervals covered by digest tables in both approaches can be expressed as

$$t_y \ = \ \frac{c \times n}{P_l} \times t_a \ , \tag{14}$$

where t_y and t_a denote the average time interval covered by a digest table in HIT and in SPIE, respectively.

At a given time, there are n digest tables recording traffic in HIT, whereas there is only one in SPIE. For a query time period Δt, let ND_y and ND_a denote the average number of digest tables being examined in the HIT and SPIE, respectively. We can write

$$ND_y = n \times \left\lceil \frac{\Delta t}{t_y} \right\rceil \ , \tag{15}$$

$$ND_a = \left\lceil \frac{\Delta t}{t_a} \right\rceil = \left\lceil n \times \frac{c}{P_l} \times \frac{\Delta t}{t_y} \right\rceil \ . \tag{16}$$

From Theorem 1 in the Appendix, we have

$$\begin{cases} ND_y \leq ND_a \ , & \text{when } \Delta t \geq \frac{P_l}{c-P_l} \times t_y \ ; \\ ND_y \geq ND_a \ , & \text{when } \Delta t \leq \frac{P_l}{c} \times t_y \ . \end{cases} \tag{17}$$

From (6), we can assume $P_l = 0.5$, then we have $c/P_l \geq 2$. Therefore, by Theorem 2 in the Appendix, we obtain a more precise conclusion:

$$\begin{cases} ND_y \leq ND_a \ , & \text{when } \Delta t \geq \frac{1}{2c} \times t_y \ ; \\ ND_y \geq ND_a \ , & \text{when } \Delta t \leq \frac{1}{2c} \times t_y \ . \end{cases} \tag{18}$$

That means the number of digest tables being examined is dependent on the time interval covered by digest tables. At high-speed routers where digest tables store short periods of traffic data, HIT examines fewer digest tables than SPIE. And at other routers, SPIE performs better.

3) Traceback Accuracy: Bloom filter is a space-efficient data structure to represent a set and check for the membership of an element in the set. When checking membership, Bloom filters never yield a false negative but may produce false positives. The false positive rate of a Bloom filter is dependent on the size of the Bloom filter and the size of the set stored. Suppose a Bloom filter is of s bits and stores e elements, the false positive rate is exponentially dependent on the value of e/s [25], which is called *memory efficiency factor* [9]. Hence, the false positive rate of a Bloom filter can be controlled by carefully choosing its memory efficiency factor.

If a router examines d digest tables for a packet digest and the false positive rate of each table is f, then the probability to get false positive results is

$$P_{f,d} = 1 - (1-f)^d. \tag{19}$$

During the traceback process, queried routers may return false positive responses, introducing spurious branches arising from the attack path. We refer to those spurious branches as *false positive branches*, or FPB. In HIT, FPBs may be generated in two cases, as illustrated in Fig. 8. One case is that FPBs arise from logging routers, the other case is that FPBs arise from marking routers. Suppose, in Fig. 8, router

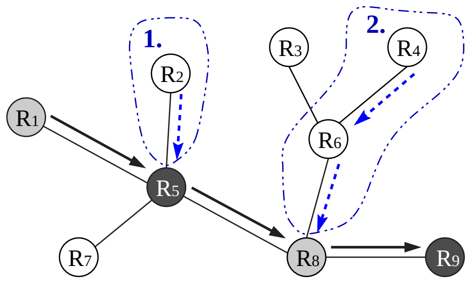

Fig. 8. False positive branches generated during the traceback process in HIT. Solid arrows represent the attack path; dashed arrows represent false positive branches. Among the routers on the attack path, R_5 and R_9 logged the attack packet, the others did not.

R_8 is the furthest router having been identified so far on the attack path. R_8 did not log but marked the attack packet. The traceback server queries the neighboring routers of R_8.

1) The queried router is on the attack path (e.g., R_5). The router logged the attack packet. For each neighbor in turn, the router embeds the ID number of that router into the attack packet, computes the packet digest, and examines relevant digest tables. The false positive results of examining those tables will graft that neighbor router (e.g., R_2) onto the attack path, introducing a FPB.
2) The queried router is not on the attack path (e.g., R_6). The false positives resulting from examining the digest tables at the router will graft that router itself onto the attack path, introducing a FPB.

Suppose, on average, each router in the network has n neighboring routers and examines ND_y digest tables when being queried. In case 1 above, the probability of a FPB is

$$P_{b_1} = 1 - (1-f)^{ND_y/n}. \tag{20}$$

In case 2 above, the probability of a FPB is

$$P_{b_2} = 1 - (1-f)^{ND_y}. \tag{21}$$

Consider the scenario where the traceback server queries the neighboring routers of a router, say R_j, on the attack path. The traceback server queries totally $n-1$ routers. Among them, one router is the upstream router on the attack path, say R_i; the other $n-2$ routers are not on the attack path. R_i examines digest tables to check which one of its $n-1$ neighbors is on the attack path. Let random variables A and B represent the number of FPBs generated when querying R_i and the other $n-2$ routers, respectively. Both A and B follow the binomial distribution. A is with parameters $n-2$ and P_{b_1}, and B is with parameters $n-2$ and P_{b_2}.

In HIT, a packet traversing the network is generally logged at every other router on the path. Let NF_y denote the average number of FPBs generated during the traceback process in HIT. For a given attack path of h hops, we have

$$\begin{aligned} NF_y &= \frac{h}{2} \times (E[A] + E[B]) \\ &= \frac{h}{2} \times ((n-2) \times P_{b_1} + (n-2) \times P_{b_2}) \\ &= \frac{h}{2} \times (n-2) \times (P_{b_1} + P_{b_2}) \,. \end{aligned} \tag{22}$$

If we assume that each router ensures $P_{b_2} \le q$, because $P_{b_1} \le P_{b_2}$, then we can get an upper bound on NF_y as

$$NF_y \le h \times n \times q \,. \tag{23}$$

In SPIE, FPBs may be generated only when querying routers not on the attack path. Suppose, on average, each router has n neighbor and examines ND_a digest tables when queried. Then the probability of a FPB is

$$P_{b'} = 1 - (1-f)^{ND_a} \,. \tag{24}$$

Let NF_a denote the average number of FPBs generated during the traceback process in SPIE. Since a packet traversing the network is logged at every router on the path, for a given attack path of h hops, we have

$$NF_a = h \times (n-2) \times P_{b'} \,. \tag{25}$$

Suppose each router ensures $P_{b'} \le q$, then we can get the same upper bound on NF_a as on NF_y:

$$NF_a \le h \times n \times q \,. \tag{26}$$

If each queried router could limit the possibility of reporting FPBs by the same value in both HIT and SPIE, these two approaches will have the same upper bound on the average number of FPBs.

FPBs result in inaccurate traceback results. However, some mechanisms may be used to prune FPBs, thereby compensating for the inaccuracy to some extent. One is to stick with the longest path and remove other shorter branches. Another is to utilize routing information to detect and prune FPBs. For example, if a router R_i is considered having forwarded the attack packet to another router R_j, we should be able to find the corresponding entry (i.e., the packets destined to the victim will be forwarded to R_j) in the routing table at R_i.

4) Simulations: We conduct simulations to supplement some of the analytic results. We focus on packet logging overhead and traceback process overhead characteristics.

For the packet logging overhead, we design simulations to study the probability that packets are logged at routers in HIT. The simulations are based on two network topologies: The first one is a synthetic transit-stub topology with one transit and 48 stub networks. The transit network includes 16 nodes which we refer to as core routers. The overall topology has 256 nodes and 353 links. The second topology is AT&T POP-level topology collected by Rocketfuel [26], which includes 115 nodes and 148 links. 19 out of 115 nodes have more than 2 neighbors. We refer to these 19 nodes as core routers.

We conduct simulations on Network Simulator (ns-2) [27]. We assume each router is connected directly with an end host. Each end host sends a packet to all other hosts. None of the packets are fragmented or transformed while traversing the network. In the simulations, we consider two scenarios. In scenario 1, all packets are logged at the first router on their paths; in scenario 2, all packets are not logged at the first router on their paths. For each scenario, we collect the number of packets logged and the number of packets forwarded by each router, calculate the probability that packets are logged at each router, and compute the cumulative distribution function of those logging probabilities.

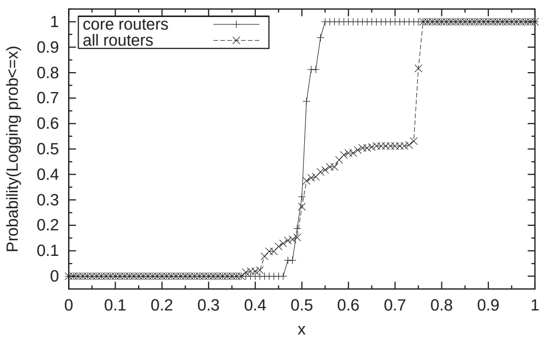

Fig. 9. The cumulative distribution function of logging probability. (Transit-stub topology. Packets are logged at the first router on the path.)

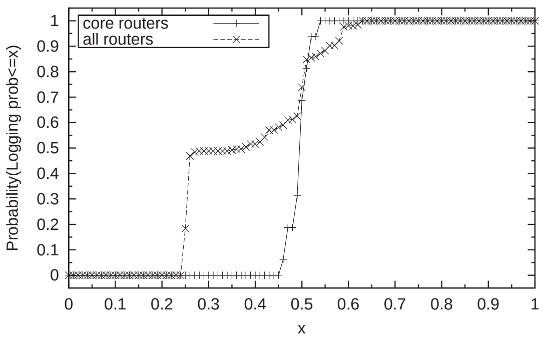

Fig. 10. The cumulative distribution function of logging probability. (Transit-stub topology. Packets are not logged at the first router on the path.)

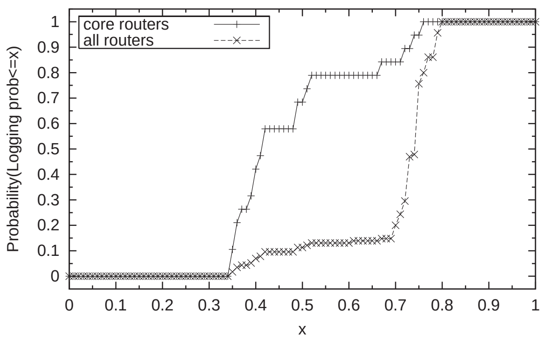

Fig. 12. The cumulative distribution function of logging probability. (AT&T topology. Packets are logged at the first router on the path.)

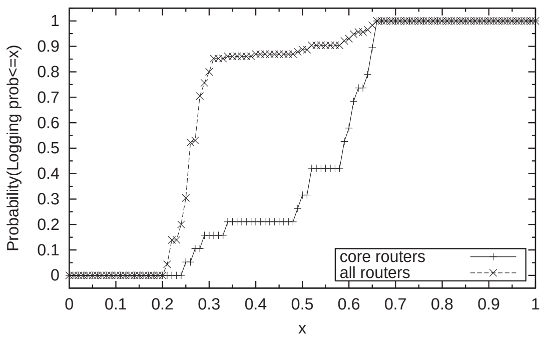

Fig. 13. The cumulative distribution function of logging probability. (AT&T topology. Packets are not logged at the first router on the path.)

Figs. 9 and 10 show the results on the transit-stub topology, in scenario 1 and 2, respectively. In both scenarios, the logging probabilities at core routers fall in the range of 45% to 55%. When considering the logging probabilities at all routers, we notice that the logging probabilities vary remarkably in different scenarios. In scenario 1, around 30% routers has a logging probability less than 50%; in scenario 2, around 70% routers has a logging probability less than 50%.

This is because the logging probabilities at the routers at or near the network edge depend significantly on whether the first router on the path logs packets or not. Consider router R_1 in Fig. 11. All traffic through R_1 is originated from or destined to H_1. In scenario 1, all ingress packets are logged at R_1, and generally half of egress packets are logged at R_1. This results in a logging probability of about 75%. In scenario 2, only around half of egress packets are logged at R_1, resulting in a logging probability of about 25%. A complementary situation applies to router R_2 in Fig. 11. Most traffic through R_2 is originated from or destined to end hosts 2 hops away (i.e., $H_3, H_4, \ldots, H_n$). In scenario 1, only a small portion of ingress packets and around half of egress packets are logged at R_2, resulting in a low logging probability. In scenario 2, most of ingress packets, around half of egress packets, and all packets between hosts of $H_3, H_4, \ldots, H_n$ are logged at R_2, resulting in a high logging probability.

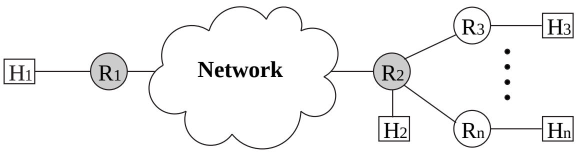

Fig. 11. Routers at or near the edge of network.

Figs. 12 and 13 show the results on AT&T topology, in scenario 1 and 2, respectively. In scenario 1, around 75% of core routers and 15% of all routers have a logging probability less than 50%. In scenario 2, around 25% of core routers and 85% of all routers have a logging probability less than 50%. We think the reason is the topology of AT&T's network. AT&T's network topology includes hubs in major cities and spokes that fan out to smaller cities. These hubs (we call them core routers in the simulations) are like router R_2 in Fig. 11, and the spokes are like router R_1 in Fig. 11. That explains why the hubs and spokes have the logging probabilities being kind of complementary to each other.

The simulation results demonstrate that 45% to 55% percent forwarded packets need to be logged at the backbone routers of a network. This confirms the analytic results of the digest table storage and access time overhead. The simulation results also show that, at the routers at or near the edge of a network, the percentage of packets which need to be logged depends on whether the first router on the network path logs packets or not. ISPs may attain a better performance through properly setting up the behavior of the edge routers.

For the traceback process overhead, we simulate the number of routers being queried during the process of tracing an attack

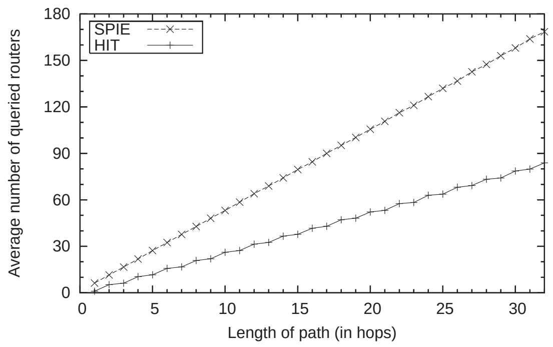

Fig. 14. The number of routers being queried during traceback process.

path of various length. The degree of each router on the path is determined randomly according to the degree distribution of a router-level topology from CAIDA (ITDK0304 skitter data) [28]. We conduct 1000 test runs for each path length. The simulation results in Fig. 14 show that HIT queries around one half as many routers as SPIE.

B. Comparison to Hybrid IP Traceback Approaches

In this section, we compare HIT with DLLT and PPPM approaches [12] based on several performance metrics.

1) Number of Packets Required for Traceback: HIT needs only one packet to construct an attack path. In general, both DLLT and PPPM need multiple packets to construct an attack path. The number of packets depends on the marking probability q at routers and the length of the attack path. In an extreme case where $q = 1$, DLLT can trace a single packet.

2) Marking Overhead on Packets: IP protocol header does not have a field provisioned for storing packet marking information. It is widely accepted in PPM approaches to overload the 16-bit IP identification field, with the price of backward incompatibility with fragmented IP traffic [7]. Although the 13-bit fragment offset field becomes meaningless when the IP identification field is overloaded, it is a challenging task to overload the fragment offset field. Since receivers regard the IP packets with non-zero fragment offset values as IP fragments [29], overloading the fragment offset field will collide with all the IP traffic in the Internet both non-fragmented and fragmented. Additional mechanisms must be in place when the fragment offset field is reused for packet marking [19].

In HIT, the marking value is stored in the 16-bit IP identification field. As mentioned in Section III-E, HIT does not mark IP fragments, thereby is backward compatible. In DLLT, the size of marking field is 34 bits and in PPPM it is 57 bits.

3) Storage Overhead: In HIT and DLLT, when logging a packet, the router records both the packet digest and the marking information carried by the packet for potential traceback process. Hence, in HIT and DLLT, the storage overhead at a router is proportionally dependent on (1) the logging probability (the percentage of the packets logged at the router), (2) total incoming link capacity, and (3) the time period for which the packet logging information are kept at the router.

In both approaches, packet digests are stored into the digest table implemented with a Bloom filter. The difference is at the storage of the marking information. In HIT, the marking information is encoded into the packet digest and does not incur additional storage overhead; while in DLLT, the 34-bit marking information is stored into a separate marking information table (MIT). For a digest table of s bits, the corresponding MIT table is of $34 \times s$ bits. If we assume that both approaches maintain the digest tables of the same size and of the same memory efficiency factor r, the storage overhead for DLLT is 34 times higher than HIT.

We consider a router with a total link capacity of b packets per unit time and with a memory efficiency factor of digest table r. As shown in Section IV, the logging probability in HIT, P_l, is around 50% on average. In DLLT, the logging probability is the same as the marking probability, so we use q to represent both probabilities in DLLT. Let S_h and S_d denote the storage overhead of per unit time in the HIT and DLLT approach, respectively. We can write

$$S_h = P_l \times b \times \frac{1}{r} = \frac{b}{2r} \,, \tag{27}$$

$$S_d = q \times b \times \frac{1}{r} \times (1 + 34) = \frac{35 \times q \times b}{r} \,. \tag{28}$$

When $q > \frac{1}{70}$, we have $S_d > S_h$. The authors in [12] propose an optimization in terms of sharing an MIT table among multiple digest tables to reduce the storage overhead of DLLT. With this sharing, they try to utilize the unused entries in the MIT table. The perfect utilization of the MIT table requires collision free mapping between the digest tables and the MIT table. In such a best case scenario, the storage overhead of DLLT becomes

$$S_d = q \times b \times (\frac{1}{r} + 34) \,. \tag{29}$$

If we use $r = 0.2$ (from [9]), the relation between S_h and S_d becomes $S_d > S_h$ for $q > 0.064$.

In PPPM, routers store the marking information in a per destination buffer. Routers also employ a Bloom filter to improve the lookup speed of the marking buffer. Thus, the storage overhead at a router is fixed and it includes the Bloom filter and a storage space of size 57×2^{32} bits for the marking buffer. The authors in [12] make an observation that the number of destinations seen by a router during a small window of time is limited. They use this observation to reduce the size of the marking buffer from 2^{32} entries to 2^a entries where a represents the size of the IP destination address suffix used to index the buffer. Under this assumption, which may cause traceback inaccuracies due to possible suffix collisions, the storage overhead of PPPM is given by the sum of the storage space for a Bloom filter, s, and the storage space of size 57×2^a bits for the marking buffer.

4) Router Processing Overhead: IP traceback schemes introduce processing overhead onto the routers in two phases (1) while creating audit trails on network traffic and (2) while conducting traceback to find out an attack path. During a peace time, routers spend processing power for the first phase and during an attack they spend processing power for the second phase. Recall that packet marking is inexpensive as it can be done in hardware. On the other hand, packet logging incurs more processing overhead on the routers.

First, we compare the three approaches based on the average overhead per router during the first phase. In HIT, a router marks 100% and logs 50% of the traffic. In both DLLT and PPPM, the percentage of the traffic being marked and logged at a router is indicated by the the marking/logging probability q, with a typical value of $q \in [0.05, 0.3]$ [12].

Next, we compare the three approaches based on the overhead per traceback process. In HIT, the processing overhead depends on (1) the number of routers and (2) number of neighbors of each router on the attack path. If an attack path has h routers, each of which has n neighbors on average, the number of routers involved during a traceback operation is given by $(n-1) \times h/2$ for HIT. In DLLT, the processing overhead depends on the number of routers on the attack path. Finally, PPPM does not introduce any processing overhead on the routers as the traceback process is conducted locally at the victim. Recall that the additional processing overhead incurred in HIT is required to traceback a single packet which is not possible in DLLT (unless $q = 100\%$) and PPPM approaches.

V. Discussion

A. Deployment

The effectiveness of log-based IP traceback increases greatly with the widespread deployment of traceback-enabled routers in the network. Similar to SPIE, it is likely that hybrid single-packet IP traceback does not require all routers to be traceback-enabled. All traceback-enabled routers within a network can be regarded as an overlay network. If the traceback server has the topology knowledge of that overlay network and each traceback-enabled router knows its overlay neighbors, the hybrid approach still works.

Tracing a packet which has traversed multiple autonomous systems (ASes) requires cooperation and trustworthiness among those ASes. Moriarty [30] proposed an inter-AS communication protocol to facilitate the cooperation among ASes during the inter-AS traceback process. Any modification of routers cannot be deployed simultaneously, or be finished in short term throughout the Internet. It is unrealistic to expect all ASes begin to deploy IP traceback services at the same time. The traceback process may halt prematurely because of the lack of support from some AS on the attack path. Korkmaz *et al.* [31] proposed a scheme for conducting the traceback process under the environment where hash-based IP traceback is partially deployed at AS-level. Because of the similarity between the hash-based approach and the hybrid approach, similar results apply to the effectiveness of the hybrid approach in AS-level partial deployment scenario.

B. Security Concerns

Attackers may write a forged marking value into attack packets. This only helps the attacker to prefix a false router to the attack path. Since a packet is marked by each and logged by every other router, the attacker cannot introduce an arbitrary attack path. In order to successfully exploit this vulnerability, (1) attackers have to know the ID numbers of the neighboring routers of the first router on the path and (2) attack packets need to enter the network at a router port which is not (or cannot be) upgraded to mark packets as described in Section III-B. We think that it is not easy to exploit this vulnerability. Even succeeding in that, attackers can prefix at most one router to the attack path. To deal with this vulnerability, the traceback server may refer to routing information for the authenticity of the furthest router on the attack path.

Attackers may flood the victim with IP fragments for the purpose of consuming more storage space and reducing the time duration for which packet digests are kept at routers. At low-speed routers where the storage space is not a concern, routers can dedicate more storage space for the digests of IP fragments. At high-speed routers where the storage space is a concern, attack traffic consumes only a small fraction of the bandwidth. Hence, the increased storage overhead at high-speed routers is trivial compared to the total amount of memory dedicated for packet digests.

VI. Conclusion

Tracing a single IP packet back to its origin is the ultimate goal of IP traceback. SPIE illustrates the feasibility of tracing individual packets with packet logging. However, the storage overhead and access time requirement for recording packet digests are fairly high at high-speed routers. On the other hand, the traceback approach based on packet marking incurs little overhead at routers, though it can only trace large packet flows.

In this paper, we have proposed a hybrid single-packet IP traceback approach based on both packet logging and packet marking. The main idea is to accumulate the information of multiple routers on the network path through packet marking, and log these accumulated path information at some of the routers on the path. Based on this idea and the current Internet environment, we have developed a concrete IP traceback approach. Our approach has the same single-packet traceback ability as SPIE, but incurs less overhead at routers. Specifically, our approach (1) reduces the storage overhead to one half, and (2) reduces the access time requirement by a factor of the number of neighboring routers.

References

[1] D. Moore, G. Voelker, and S. Savage, "Inferring Internet denial of service activity," in *Proc. of USENIX Security Symposium*, (Washington, DC, USA), August 2001.

[2] P. Vixie, G. Sneeringer, and M. Schleifer, "Event report: Events of 21-Oct-2002." `http://d.root-servers.org/october21.txt`.

[3] D. Pappalardo and E. Messmer, "Extortion via DDoS on the rise," *Network World*, May 16, 2005. `http://www.networkworld.com/news/2005/051605-ddos-extortion.html`.

[4] A. Hussain, J. Heidemann, and C. Papadopoulos, "A framework for classifying denial of service attacks," in *Proc. of ACM SIGCOMM*, (Karlsruhe, Germany), August 2003.

[5] H. Burch and B. Cheswick, "Tracing anonymous packets to their approximate source," in *Proc. of USENIX Systems Administration Conference*, (New Orleans, LA, USA), December 2000.

[6] G. Sager, "Security fun with OCxmon and cflowd." Presentation at the Internet2 working group meeting, November 1998. `http://www.caida.org/funding/ngi/content/security/1198/`.

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

[8] H. Lipson, "Tracking and tracing cyber-attacks: Technical challenges and global policy," tech. rep., Software Engineering Institute, Carnegie Mellon University, November 2002.

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

[10] B. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[11] C. Gong and K. Sarac, "IP traceback based on packet marking and logging," in *Proc. of IEEE ICC*, (Seoul, Korea), May 2005.

[12] B. Al-Duwairi and G. Manimaran, "Novel hybrid schemes employing packet marking and logging for IP traceback," *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, no. 5, pp. 403–418, 2006.

[13] A. Snoeren, C. Partridge, L. Sanchez, C. Jones, F. Tchakountio, S. Kent, and W. Strayer, "Hash-based IP traceback," in *Proc. of ACM SIGCOMM*, (San Diego, CA, USA), August 2001.

[14] T. Doeppner, P. Klein, and A. Koyfman, "Using router stamping to identify the source of IP packets," in *Proc. of ACM Conf. on Computer and Communications Security*, (Athens, Greece), November 2000.

[15] D. Song and A. Perrig, "Advanced and authenticated marking schemes for IP traceback," in *INFOCOM*, (Anchorage, AK, USA), April 2001.

[16] D. Dean, M. Franklin, and A. Stubblefield, "An algebraic approach to IP traceback," *ACM Trans. on Info. and System Security*, vol. 5, no. 2, pp. 119–137, 2002.

[17] M. Goodrich, "Efficient packet marking for large-scale IP traceback," in *Proc. of ACM CCCS*, (Washington, DC, USA), November 2002.

[18] A. Yaar, A. Perrig, and D. Song, "FIT: Fast Internet traceback," in *Proc. of IEEE INFOCOM*, (Miami, FL, USA), March 2005.

[19] C. Gong and K. Sarac, "Toward a more practical marking scheme for IP traceback," in *Proc. of BROADNETS*, (San Jose, CA, USA), October 2006.

[20] T. Lee, W. Wu, and W. Huang, "Scalable packet digesting schemes for IP traceback," in *Proc. of IEEE ICC*, (Paris, France), June 2004.

[21] J. Li, M. Sung, J. Xu, L. Li, and Q. Zhao, "Large-scale IP traceback in high-speed Internet: Practical techniques and theoretical foundation," in *Symposium on Security and Privacy*, (Oakland, CA, USA), May 2004.

[22] M. Muthuprasanna, G. Manimaran, M. Manzor, and V. Kumar, "Coloring the internet: Ip traceback," in *Proc. IEEE Intl. Conference on Parallel and Distributed Systems*, (Minneapolis, MN, USA), July 2006.

[23] S. McCreary and K. Claffy, "Trends in wide area IP traffic patterns: A view from Ames Internet exchange," in *ITC Specialist Seminar on IP Traffic Modeling, Measurement and Management*, (Monterey, CA, USA), September 2000.

[24] I. Stoica and H. Zhang, "Providing guaranteed services without per flow management," in *SIGCOMM*, (Cambridge, MA, USA), August 1999.

[25] A. Broder and M. Mitzenmacher, "Network applications of Bloom filters: A survey," *Internet Mathematics*, vol. 1, no. 4, pp. 485–509, 2005.

[26] Rocketfuel. `http://www.cs.washington.edu/research/networking/rocketfuel/`.

[27] Network Simulator (ns-2). `http://www.isi.edu/nsnam/ns/`.

[28] CAIDA. `http://www.caida.org/tools/skitter/`.

[29] J. Postel, "Internet protocol." RFC 791, September 1981.

[30] K. M. Moriarty, "Incident handling: Real-time inter-network defense." Internet draft, October 2005. `http://www.ietf.org/internet-drafts/draft-ietf-inch-rid-05.txt`.

[31] T. Korkmaz, C. Gong, K. Sarac, and S. Dykes, "Single packet IP traceback in AS-level partial deployment scenario," *International Journal of Security and Networks*, vol. 2, no. 1/2, pp. 95–108, 2007.

APPENDIX

Theorem 1: Let $f(x) = n\lceil x \rceil$ and $g(x) = \lceil nbx \rceil$, where $n \in \mathbb{N}$ and $n \geq 1$; $b \in \mathbb{R}$ and $b > 1$; $x \in \mathbb{R}$ and $x > 0$. Then,

$$\begin{cases} f(x) \leq g(x)\ , & \text{when } x \geq \frac{1}{b-1}\ ; \\ f(x) \geq g(x)\ , & \text{when } x \leq \frac{1}{b}\ . \end{cases}$$

Proof: **(1)** When $x \geq \frac{1}{b-1}$, we have

$$(b-1)x \geq 1 \ \Rightarrow\ n(b-1)x \geq n \ \Rightarrow$$

$$nbx - nx \geq n \ \Rightarrow\ nbx \geq n(x+1)\ .$$

Because $f(x) = n\lceil x \rceil < n(x+1)$ and $g(x) = \lceil nbx \rceil \geq nbx$, so $f(x) \leq g(x)$.

(2) When $x \leq \frac{1}{b}$, because $b > 1$, so $0 < x < 1$. Thus

$$f(x) = n\lceil x \rceil = n\ .$$

On the other hand, $nbx \leq nb\frac{1}{b} = n$. Thus

$$g(x) = \lceil nbx \rceil \leq \lceil n \rceil = n\ .$$

Therefore, $f(x) \geq g(x)$. ■

Theorem 2: Let $f(x) = n\lceil x \rceil$ and $g(x) = \lceil nbx \rceil$, where $n \in \mathbb{N}$ and $n \geq 1$; $b \in \mathbb{R}$ and $b \geq 2$; $x \in \mathbb{R}$ and $x > 0$. Then,

$$\begin{cases} f(x) \leq g(x)\ , & \text{when } x \geq \frac{1}{b}\ ; \\ f(x) \geq g(x)\ , & \text{when } x \leq \frac{1}{b}\ . \end{cases}$$

Proof: **(1)** When $x \geq \frac{1}{b}$, there are three cases to consider:
Case 1: $x \in [\frac{1}{b}, 1)$. In this case, $\lceil x \rceil = 1$ and $nbx \geq n$. Thus,

$$f(x) = n\lceil x \rceil = n\ ,$$

$$g(x) = \lceil nbx \rceil \geq \lceil n \rceil = n\ .$$

Case 2: $x = m$ ($m \in \mathbb{N}$ and $m \geq 1$). In this case, $\lceil x \rceil = m$ and $bx = bm \geq 2m > m$. Thus,

$$f(x) = n\lceil x \rceil = nm\ ,$$

$$g(x) = \lceil nbx \rceil = \lceil nbm \rceil \geq \lceil nm \rceil = nm\ .$$

Case 3: $x = m + y$ ($m \in \mathbb{N}$ and $m \geq 1; y \in (0,1)$). In this case, $\lceil x \rceil = m + 1$ and $bx > bm \geq 2m \geq m + 1$. Thus,

$$f(x) = n\lceil x \rceil = n(m+1)\ ,$$

$$g(x) = \lceil nbx \rceil \geq \lceil nbm \rceil \geq \lceil n(m+1) \rceil = n(m+1)\ .$$

Putting together the 3 cases above, we have $f(x) \leq g(x)$.

(2) When $x \leq \frac{1}{b}$, by Theorem 1, we have $f(x) \geq g(x)$. ■

PLACE PHOTO HERE

Chao Gong received his Ph.D. in computer science from the University of Texas at Dallas in 2007. He is now an Assistant Professor in the Department of Computer Science at University of Mary Hardin-Baylor, Belton, TX. His research has focused on the security and management of computer networks.

PLACE PHOTO HERE

Kamil Sarac received the M.S. and Ph.D. degrees in computer science from the University of California at Santa Barbara, in 1997 and 2002, respectively. He is currently an Assistant Professor in the Department of Computer Science at the University of Texas, Dallas. His research interests include computer networks and protocols; network and service monitoring and Internet measurements; overlay networks and their use in network security and denial-of-service defense; and multicast communication.