

TOPO: A Topology-aware Single Packet Attack Traceback Scheme

Linfeng Zhang and Yong Guan
Department of Electrical and Computer Engineering
Iowa State University
Ames, Iowa 50011
{zhanglf, yguan}@iastate.edu

Abstract—With the phenomenal growth of the Internet, more and more people enjoy and depend on its provided services. Unfortunately, the number of network-based attacks is also increasing quickly. Network attackers can very easily hide their identities, and thereby reduce the chance of being captured and punished. Some attacks can even succeed by using only one or a few well-targeted packets. Therefore, it is desirable to design effective and efficient single packet IP traceback systems to attribute attackers. Several single packet IP traceback systems have been designed using Bloom filters. However, the inherent false positives of Bloom filters caused by unavoidable collisions restrain the effectiveness of these systems.

To reduce the impact of unavoidable collisions in Bloom filters, we propose a topology-aware single packet IP traceback system, namely *TOPO*. We utilize router’s local topology information, i.e., its immediate predecessor information. Our performance analysis shows that *TOPO* can reduce the number and scope of unnecessary queries, and significantly decrease false attributions. Furthermore, to improve the practicability of Bloom filter-based IP traceback systems, we design *TOPO* to allow partial deployment while maintaining its traceback capability. When Bloom filters are used, it is difficult to decide their optimal control parameters a priori. We design a k -adaptive mechanism which can dynamically adjust parameters of Bloom filters to reduce the false positive rate.

Index Terms—Attack Attribution, IP Traceback, Partial Deployment, Bloom Filter, Network Security

I. INTRODUCTION

With the phenomenal growth of the Internet, more and more people enjoy and depend on the convenience of its provided services. Unfortunately, the number of network-based attacks is also increasing very quickly. The consequences are serious and, increasingly, financial disastrous, but ironically, only few of the attackers have been captured and thereby punished because of the stateless nature of the Internet. Network-based attackers can easily hide their identities through IP spoofing, stepping stones, network address translators (NATs), Mobile IP or other ways, and thereby reduce the chance of being captured. The current IP network infrastructure lacks measures which can effectively deter and identify motivated and well-equipped attackers. As a result, due to the lack of effective attack attribution techniques and concerns for negative publicity, the percentage of organizations reporting computer intrusions to law enforcement has continued its multi-year decline [14]. Therefore, it is desirable to design effective and efficient IP traceback systems to attribute attackers and help to reconstruct cyber crime scenes.

Building systems that can reliably trace attack packets back to their real origins in the current IP networks is a first and important step in making cyber criminals accountable. There are many forms of network-based attacks. While the most-widely reported DDoS attacks are conducted by flooding networks with large amounts of traffic, there are network-based attacks that require significantly smaller packet flows. Some attacks (e.g., Teardrop) can even succeed by using only one well-targeted packet. In building such traceback systems, there are a number of significant challenges including

which packets to trace, how to trace *long-lifetime*¹ and *short-lifetime*² attacks, and how to minimize router processing overhead and storage space requirements, while satisfying applications’ and network users’ privacy requirements.

Several IP traceback schemes have been designed to attribute the origin of IP packets through the Internet. We roughly categorize them into four primary classes: (i) *Active Probing* [10], [28], (ii) *ICMP Traceback* [7], [19], [30], (iii) *Packet Marking* (including deterministic, probabilistic, and algebraic packet marking) [6], [12], [22], [23], [26], and d) *Log-based Traceback* [17], [20], [24], [25]. The IP traceback systems built atop of approaches of Active Probing, ICMP traceback and Packet Marking require a sufficiently large set of attack packets to make traceback possible, which are not effective to traceback short-lifetime or single packet attacks. Log-based traceback schemes seem suitable to attribute individual packet to its origin. However, several log-based methods such as [20] require recording the entire network traffic for future attack traceback. Obviously, such methods have overly high storage space requirements, which make them impractical to be used for current high-speed IP networks, especially those with heavy traffic.

Thereafter, to address the problem of log-based IP traceback systems’ overly large storage space requirement, two IP traceback systems [24], [25] were designed using Bloom filters [8], a space-efficient data structure for representing a set of elements to respond membership queries. However, although Bloom filters are space-efficient data structures in responding membership queries, they have inherent unavoidable collisions which produce false positives, and thus restrain the effectiveness of these systems.

In this paper, we propose a Bloom filter-based topology-aware single packet IP traceback system, namely *TOPO*, which utilizes router’s local topology information, i.e., its immediate predecessor information, to traceback. Our theoretical analysis and experimental evaluation show that *TOPO* can significantly reduce the number and scope of unnecessary queries and thus, significantly decrease the false attributions to innocent nodes.

Furthermore, we consider the practicability of Bloom filter-based IP traceback systems. In some real world networks, it is difficult or even impossible to install an IP traceback system on all the routers on the network. Therefore, partial deployment is an important and desired property when designing and implementing IP traceback systems. We analyze and show that *TOPO* is suitable to be partially deployed while maintaining its traceback capability.

Finally, we discuss an issue on utilizing Bloom filters which has

¹By *long-lifetime*, we mean that there are a sufficiently large number of attack packets available for traceback systems.

²By *short-lifetime*, we mean that there are a significantly smaller number of attack packets available for traceback systems. Some attacks might only use a single packet.

never been studied. When Bloom filters are applied in IP traceback systems, it is difficult to decide their optimal control parameters *a priori* and thus achieve the lowest false positive rate. Based on our analytical results, we design a k -adaptive mechanism to dynamically adjust parameters of Bloom filters such that our IP traceback system can achieve the best performance in terms of false attribution rates and storage space requirement.

The rest of this paper is structured as follows: Section II discusses the related work. We articulate the problems and our goals in Section III. We propose our topology-aware single packet IP traceback system TOPO in Section IV. The proposed system is analyzed and evaluated in Section V. Several interesting design issues are discussed in Section VI. We finally conclude in Section VII with a summary and suggestions for future work.

II. RELATED WORK

In this section, we review major existing IP traceback schemes that have been designed to attribute the origin of IP packets through the Internet. We roughly categorize them into four primary classes: a) Active Probing [10], [28], b) ICMP Traceback [7], [19], [30], c) Packet Marking (including deterministic, probabilistic, and algebraic packet marking) [6], [12], [22], [23], [26], and d) Log-based Traceback [17], [20], [24], [25].

A. Active Probing

Stone [28] proposed a traceback scheme called *CenterTrack*, which selectively reroutes packets in question directly from edge routers to some special tracking routers. The tracking routers determine the ingress edge router by observing from which tunnel the packet arrives. This approach requires the cooperation of network administrators and the management overhead is considerably large.

Burch and Cheswick [10] outlined a technique for tracing spoofed packets back to their actual source without relying on the cooperation of intervening ISPs. The victim actively changes the traffic in particular links and observes the influence on attack packets, and thus can determine where the attack comes from. This technique cannot work well on distributed attacks, and requires the attacks remain active during the time period of traceback.

B. ICMP Traceback (*iTrace*)

Bellovin [7] proposed a scheme named *iTrace* to traceback using ICMP messages for authenticated IP marking. In this scheme, each router samples (with low probability) the forwarding packets, copies the contents into a special ICMP traceback message, adds its own IP address as well as the IP of the previous and next hop routers, and forwards the packet either to the source or destination address. By combining the information obtained from several of these ICMP messages from different routers, the victim can then reconstruct the path back to the origin of the attacker.

A drawback of this scheme was that it is much more likely that the victim will get ICMP messages from routers nearby than from routers farther away. This implies that most of the network resources spent on generating and utilizing *iTrace* messages will be wasted. An enhancement of *iTrace*, called "*Intention-Driven iTrace*", was proposed in [19], [30]. By introducing an extra "intention-bit", it is possible for the victim to increase the probability of receiving *iTrace* messages from remote routers.

C. Packet Marking

Savage et al. [23] proposed a *Probabilistic Packet Marking* (PPM) scheme. Thereafter, several other PPM-based schemes have been developed [26], [22], [12]. The baseline idea of PPM is that routers

probabilistically write partial path information into the packets during forwarding. If the attacks are made up of a sufficiently large number of packets, eventually, the victim may get enough information by combining a modest number of marked packets to reconstruct the entire attack path. This allows victims to locate the approximate source of attack traffic without requiring outside assistance.

Deterministic Packet Marking (DPM) scheme proposed by Belenky and Ansari [6] involves the marking of each individual packet when it enters the network. The packet is marked by the interface closest to the source of the packet on the edge ingress router. The mark remains unchanged as long as the packet traverses the network. However, there is no way to get the whole paths of the attacks.

Dean et al. proposed an *Algebraic Packet Marking* (APM) which reframes the traceback problem as a polynomial reconstruction problem and uses techniques from algebraic coding theory to provide robust methods of transmission and reconstruction. The advantage of this scheme is that it offers more flexibility in design and more powerful techniques that can be used to filter out attacker generated noise and separate multiple paths. But it shared similarity with PPM in that it requires a sufficiently large number of attack packets.

D. Log-based Traceback

The basic idea of log-based traceback is that each router stores the information (digests, signature, or even the packet itself) of network traffic through it. Once an attack is detected, the victim queries the upstream routers by checking whether they have logged the attack packet in question or not. If the attack packet's information is found in a given router's memory, then that router is deemed to be part of the attack path. Obviously, the major challenge in log-based traceback schemes is the storage space requirement at the intermediate routers.

Matsuda et al. [20] proposed a hop-by-hop log-based IP traceback method. Its main features are logging *packet feature* that is composed of a portion of the packet for identification purpose, and an algorithm using data-link identifier to identify the routing of a packet. However, for each received packet, about 60 bytes data should be recorded. The resulted large memory space requirement prevents this method from being applied to high speed networks with heavy traffic.

Although today's high-speed IP networks suggest that classical log-based traceback schemes would be too prohibitive because of the huge memory requirement, log-based traceback becomes attractive after Bloom filter-based (i.e., hash-based) traceback schemes were proposed. *Bloom filters* were presented by Burton H. Bloom [8] in 1970, and have been widely used in many areas such as database and networking [9]. A Bloom filter is a space-efficient data structure for representing a set of n elements to respond membership queries. It is a vector of m bits which are all initialized to value 0. Then each element is inserted into the Bloom filter by hashing it using k independent uniform hash functions with range $\{1, 2, \dots, m\}$ and setting the corresponding k bits (some bits may be overlapped) in the vector to value 1. Given a query whether an element is present in the Bloom filter, we hash this element using the same k hash functions and check if all the corresponding bits are set to 1. If any one of them is 0, then undoubtedly this element is not stored in the filter. Otherwise, we would say that it is present in the filter, although there is a certain probability that the element is determined to be in the filter while it is actually not. Such false cases are called *false positives*. The space-efficiency of Bloom filters is achieved at the cost of a small acceptable false positive rate f . From [13], we have

$$f = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^k \approx \left(1 - e^{-kn/m}\right)^k. \quad (1)$$

When m and n are given, f is minimized for

$$k = \ln 2 \times m/n, \quad (2)$$

Thus, we have

$$f = 2^{-k} \approx 0.6185 \frac{m}{n}. \quad (3)$$

Bloom filters were first introduced into IP traceback area by Snoeren et al. [25]. They built a system named *Source Path Isolation Engine* (SPIE) which can trace the origin of a single IP packet delivered by the network in the recent past. They demonstrated that the system is effective, space-efficient and implementable in current or next-generation routing hardware. Bloom filters are used in each SPIE-equipped router to record the digests of all packets it received in the recent past. The digest of a packet is exactly several hash values of its non-mutable IP header fields and the prefix of the payload. Strayer et al. [29] extended this traceback architecture to IP-v6.

Shanmugasundaram, et al. [24] proposed a *payload attribution system* (PAS) based on a *Hierarchical Bloom filter* (HBF). HBF is such a Bloom filter in which an element is inserted several times using different parts of the same element. Compared with SPIE which is a packet digesting scheme, PAS only uses the payload excerpt of a packet. It is useful when the packet header is unavailable.

Li et al. [17] proposed a Bloom filter-based IP traceback scheme that requires an order of magnitude smaller processing and storage cost than SPIE, thereby being able to scale to much higher link speed. The baseline idea of their approach is to sample and log a small percentage of packets, and 1 bit packet marking is used in their sampling scheme. Therefore, their traceback scheme combines packet marking and packet logging together. Their simulation results showed that the traceback scheme can achieve high accuracy, and scale well to a large number of attackers. However, as the authors also pointed out, because of the low sampling rate, their scheme is no longer capable to trace one attacker with only one packet.

III. PROBLEMS AND GOALS

A. Problems in Existing Traceback Schemes

To prevent the Internet from being attacked, it is desirable to design effective and efficient IP traceback systems to attribute the attackers and help reconstruct cyber crime scenes. *IP traceback problem* is to traceback the network path(s) the attack traverses and identify the real attackers.

Active Probing, ICMP traceback and packet marking schemes are designed to traceback those long-lifetime attacks, and they are not suitable to traceback single packet attacks or short-lifetime attacks. SPIE is designed to trace the origin of a single IP packet delivered by the network in the recent past. In SPIE, when an intrusion detection system (IDS) detects an attack, it sends out a query message to SPIE. SPIE then dispatches the query message to the relevant routers for processing. After a router receives a query about whether it has forwarded a packet with a specific arrival time, it checks its Bloom filter for that time interval. If the result is ‘present’ in the Bloom filter, the router must continue to query all its upstream neighbors. Consequently, there would be a lot of unnecessary queries sent to innocent routers. The unnecessary queries force the routers to spend CPU time and other resources to respond, and thus reduce the performance of their tasks as routers. Furthermore, these unnecessary queries can introduce innocent nodes into the attack graph because of the unavoidable false positives of Bloom filters, which in turn cause false attributions. Consequently, it is desirable to design new mechanisms to control the unnecessary queries and thus reduce the false attributions to innocent nodes.

For any proposed IP traceback systems, besides traceback effectiveness, practicability is also an important and desired property in evaluating their system performance. In some real world networks, it is difficult or even impossible to install Bloom filter-based IP traceback systems, such as SPIE [25], PAS [24] and TOPO proposed in this paper, on all routers. Although some routers in these networks may easily install and activate a Bloom filter-based IP traceback system, there are some routers such as core routers that are hardly to be updated because of the high overhead involved or other administrative issues.

If a Bloom filter-based IP traceback system can be partially deployed without losing (or compromising a little) traceback capability compared with the fully deployed system, then it will have the following potential advantages:

- **Low cost:** Such a system may avoid wasting money and labor on updating routers which are difficult and expensive to update. In addition, it is possible to reduce memory space required by the whole traceback system [18].
- **Flexibility:** Such a system may be implemented on networks which have routers that are impossible to be updated. Furthermore, it also provides flexibility to the administrator who launches traceback or investigation of a particular attack. She may enable only a portion of network routers and thereby avoid alerting the attacker.

Therefore, an IP traceback system which can be practically deployed should have the property of partial deployment.

Bloom Filters have been used in many network applications. In some applications, the element number n that should be inserted is known *a priori* before the construction of Bloom filters. For instance, Bloom filters are used to store a dictionary of unsuitable passwords for security purposes, where the number of passwords is countable [27]. With the fixed element number n , given any required false positive rate f and the memory size m , the optimal hash function number k can be calculated by equation (2). However, in IP traceback systems, the traffic volume that needs to be recorded varies significantly, especially when the network is under attacks. This indicates that the packet number n is unknown *a priori*, when m and k have to be decided in advance before traceback system is deployed. For different values of n , there are different optimal values of k . Therefore, a Bloom filter-based IP traceback system faces the problem of how to adaptively adjust parameter k to achieve a better false positive rate.

B. System Model

We consider IP networks (the Internet) where TCP/IP architecture and protocols are being used. An IP network consists of a number of host computers and network devices (e.g., routers or switches), which are connected by physical links on which packets are forwarded. We make the following assumptions in IP networks:

- Most routers are reliable.
- Some routers’ software and hardware can be updated.

C. Threat Model

With the phenomenal growth of Internet, more and more people enjoy and depend on the convenience of its provided service. Meanwhile, the number of network-based attacks is increasing very quickly. The reality is that only few of the attackers have been captured and punished due to the stateless nature of the Internet and the readily available tools and techniques on the Internet that are easily taken to conceal themselves from being discovered.

It is well-documented that many attackers use spoofed IP source addresses in their attack packets. As these packets traverse the Internet, little useful information is left for the victims to identify their true origins. Some attackers also launch their attacks behind a chain of compromised machines which are called “stepping stones”. Some attacks, such as *SYN flood Denial-of-Service* (DoS) attack, need to flood network links with large amounts of packets, while there are other attacks which require significantly smaller packet flows. Furthermore, some sophisticated attackers can start and finish their attacks using a single well-targeted packet, such as *LAND* attack [1], *Ping of Death* attack [2] and *Teardrop* attack [3].

D. Problem Definition and Our Goals

In this paper, we aim at designing a Bloom filter-based topology-aware single packet IP traceback system, namely TOPO, to solve the problems discussed above. We set the following as our goals in the design of TOPO:

- 1) To design a single packet IP traceback system which has fewer unnecessary query messages and fewer false attributions to innocent nodes.
- 2) To design a single packet IP traceback system which needs not to be fully deployed in the entire network.
- 3) To design a mechanism which helps achieve the best performance of Bloom filters by adaptively adjust parameter k .

IV. SYSTEM DESCRIPTION

In this section, we first introduce topology-aware IP traceback mechanism which is the baseline idea of TOPO, and then discuss the partial deployment issue and the design of the k -adaptive mechanism for dynamically adjusting parameters of Bloom filters to achieve better performance.

A. Topology-aware IP Traceback

We first define three terminologies used in the paper:

- **Packet Signature** is defined as the information to identify individual packet from each other.
- **Packet Predecessor** is defined as the immediate upstream neighbor which sends the packet to the current router.
- **Predecessor Identifier** is defined as the information to identify the predecessor of a packet from other upstream neighbors.

As we discussed above, we need to find a way to control the production of unnecessary queries. This can be achieved if the routers have the ability to identify which upstream routers should be queried and which else should not be queried. Therefore, if we can check not only the presence of each packet but also the packet predecessor information, we can only query the exact predecessor and thus significantly reduce the unnecessary queries. This requires that we record not only the packet signature but the predecessor information.

An intuitive way to record packet predecessor information is to separate the incoming packets into several Bloom filters, each of which only store the packets coming from a distinct predecessor. Snoeren et al. [25] discussed that a router may maintain separate Bloom filters for each of its input port, because different upstream neighbors typically use different input ports. Although this complicates the query process, the input port isolation may reduce the number of upstream neighbors that need to be queried. However, the number of predecessors (or active ports) within a certain time interval is an unknown parameter: Most routers cannot decide how many upstream neighbors will send packets to it within a time interval in a real world network. Even though we may estimate the maximum probable number of predecessors, another problem appears that how

to divide the limited memory on the router for the Bloom filters of individual predecessors. We do not know how many packets will be received from a certain predecessor. As we learn from equation (1), the false positive rate of Bloom filters depends on m/n . If we divide the memory equally, then the Bloom filter which a large amount of packets are inserted into will have high false positive rate. Therefore, it is not an effective and practical solution to divide the limited memory into several separate Bloom filters.

Another way of recording predecessor information is to create a list for each bit which is set to 1 in the Bloom filter. This list is used to record all the predecessors which have set this bit. With these lists, the presence of a packet is that all the k corresponding bits are set to 1, and there is at least one common predecessor which appears on all the k corresponding lists. However, the extra lists will consume a lot of memory, and thus degrade the Bloom filter’s performance or increase the system memory requirement.

We propose that the router still only maintains one Bloom filter at a time. The Bloom filter not only keeps the packet signature, but also the predecessor identifier. The predecessor identifier can be hashed into the Bloom filter with the packet signature together. Such an operation really can record the topology information without increasing false positive rate or requiring larger memory space.

Now we introduce TOPO which is based on the above idea. TOPO is constructed on some special routers which are equipped with Bloom filters, and we call them TOPO-equipped routers. When a packet travels through the network where TOPO is deployed, no matter whether it is an attack packet or not, the TOPO-equipped routers on its path record the packet signature and predecessor information. If an attack packet is identified by the victim, the victim’s address, packet signature, and packet arrival time, are reported to TOPO as a traceback request. TOPO then begins traceback by sending a query message to the TOPO-equipped router responsible for the region containing the victim. This router then checks its record and continues to query other TOPO-equipped routers if necessary. Finally, all responses from queried TOPO-equipped routers are gathered by TOPO to generate the attack graph. The attack graph is used for further analysis and corresponding actions.

We show the details of TOPO-equipped router’s behaviors when it records a packet and responds a query in Figure 1 and 2 respectively.³

1) Record a Packet:

- When a TOPO-equipped router receives a packet, it first extracts the packet signature and predecessor identifier.
- The predecessor identifier is inserted into an extra predecessor list if the predecessor has not been inserted before.
- The router calculates the k hash values of the combination of packet signature and predecessor identifier, and inserts them into its Bloom filter by setting the corresponding bits to 1.
- At the end of the anticipated time interval, the current Bloom filter and the predecessor list are archived by flushing the oldest ones, and a new Bloom filter and predecessor list start.

2) Respond a Query:

- When a TOPO-equipped router receives a query message, it first retrieves the Bloom filter and the predecessor list for the relevant time interval using the given attack packet arrival time in the query message.
- Each predecessor identifier on the list is combined with the packet signature provided by the query message, and the com-

³We do not discuss the topic of malicious behaviors of compromised routers, which is beyond the scope of this paper. We also do not discuss the packet transformation issue, which is well discussed in [25].

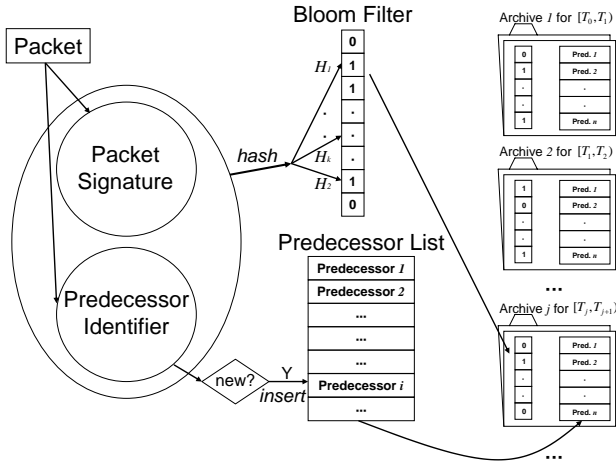


Fig. 1. Router's Behaviors when Receiving a Packet

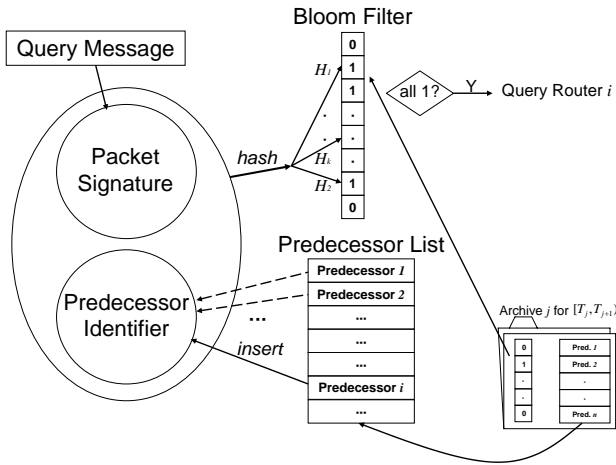


Fig. 2. Router's Behaviors when Receiving a Query Message

bination is hashed using the same k hash functions to check if it is present in the Bloom filter.

- The router will respond the query that it forwarded this packet before if there is at least one presence found. If so, it only continues to query the predecessor(s) which is (are) present in the Bloom filter.

For a router which is not TOPO-equipped, when receiving a packet, it just simply forwards it to the next hop in the path; when receiving a query message, it simply forwards it to all of its upstream neighbors.

B. Partial Deployment

Ideally we can equip TOPO on all routers on the network, and thus can trace back single packet effectively. However, many legacy routers cannot be updated, or there is no enough money to update all routers. Therefore, with resource and policy restriction, in most cases we have to partially deploy an IP trace back system. One simple partial deployment mechanism is that Bloom filter-based IP traceback system is only installed on edge routers and all internal routers remain unchanged. Under such architecture, all edge routers will be queried for any traceback request. Therefore all innocent end nodes have the possibility to be considered attackers, and thus introduce more false positives. Furthermore, there are two other drawbacks using such a

partial deployment mechanism. First, the victim cannot get a full attack graph; second, the query burden on each edge router might be very heavy. Therefore, it is necessary to deploy Bloom filters at least on part of internal routers if the full attack graph is desired or traceback requests are frequently delivered.

To partially deploy TOPO, we need to find a way to select particular routers to equip TOPO to achieve the highest traceback performance with the lowest cost. We call it *TOPO-equipped routers placement problem*. Actually, this problem should be solved as an optimization problem. For instance, given the network map, the network traffic information, the distribution of attacks, and the costs of updating routers to TOPO-equipped, we optimize the traceback system with respect to the total cost and traceback performance (i.e., a combination of cost, false positives and false negatives). When the number of candidate routers is huge, it is time-consuming to find the optimized result. Furthermore, it is hard to get the distribution of attacks before deployment of TOPO.

In this paper, we propose an intuitive but effective method to place the TOPO-equipped routers. The basic idea is that if the distribution of attacks is not available, intuitively an evenly distributed IP traceback system should have better performance. We assume that we know all the possible paths of the network, which can be achieved through some known network mapping tools [11], [15], [16]. To evenly distribute N TOPO-equipped routers on the network, we first sort all paths in descending order. We then equip TOPO on the median router of the longest path. Each path through this router is divided into two shorter paths. We sort all paths again and choose the median router of the longest path. We repeat these steps until we equip TOPO on N routers. We provide the performance analysis of partially deployed TOPO in the next section.

C. k -Adaptive Mechanism for Bloom Filters

When a new Bloom filter starts to record the arriving packets, the exact number of packets that would be inserted into this Bloom filter is unknown. As shown in formula (1), the false positive rate is decided by n and k when m is fixed. Although we can estimate a possible value of n using historical knowledge about network statistics and choose a corresponding k , in some scenarios n may dynamically change in a large range, especially when the network is under attacks. In such cases, the false positive rate will be (greatly) higher than the optimal rate, as shown in Figure 4 of [13]. However, we cannot reconstruct a Bloom filter with the optimal k after we finally know the packet number n , because when we notice that we should use a better k , the previous packets have passed because of the limited memory on routers. Therefore, we have no chance of selecting a better k and hashing all the previous packets again.

Someone may argue that when a Bloom filter is saturated, it can be archived and a new filter starts. However, when the memory is limited, there may not be extra memory space for those unexpected packets. For instance, a router in the IP traceback system is designed to trace the traffic within 1 hour with the granularity of 1 minute, and divides its memory into 61 slices (1 slice is used to store the packets in the current minute, and the other 60 slices are for these 60 archived Bloom filters in the recent past.). If the router finds that the current Bloom filter saturates after 30 seconds, it cannot archive the current Bloom filter by flushing the oldest archived filter, because such an operation makes the router fail to respond the queries of the previous packets between 59 minute 30 second and 60 minute ago, and thus violates the traceback system goals and requirements. Otherwise, after an attacker finishes a serious attack, it can easily flood the network and flush its previous attack traffic before the system is aware of its

attack and starts the traceback. Therefore, we need a mechanism to adjust the hash function number k with respect to the dynamic n to achieve the best performance of Bloom filters.

A direct solution is to construct several Bloom filters simultaneously which have different numbers of hash functions. After all elements are inserted, we archive the Bloom filter with the optimal k and throw away all others. Obviously, this solution requires much more memory for the extra Bloom filters, and thus perhaps decreases the entire performance eventually.

We propose an effective k -adaptive mechanism as shown in Figure 3 which uses a table v with m Q -bit entries to record the results of K m -bit Bloom filters with different numbers of hash functions, if every smaller hash function set is the subset of larger ones, where

$$Q = \lceil \log_2(K + 1) \rceil. \quad (4)$$

Let H_1, H_2, \dots, H_K be the K hash function sets which have k_1, k_2, \dots, k_K hash functions respectively, and $k_1 < k_2 < \dots < k_K$.

$$\begin{aligned} H_1 &= \{h_1, \dots, h_{k_1}\}, \\ H_2 &= \{h_1, \dots, h_{k_2}\}, \\ &\dots \\ H_K &= \{h_1, \dots, h_{k_K}\}. \end{aligned}$$

Therefore $H_1 \subset H_2 \subset \dots \subset H_K$. For each hash function h_i among the total k_K hash functions, let s_i denote the number of hash sets it belongs to in H_1, H_2, \dots, H_K . For instance, if $h_i \notin H_j$ and $h_i \in H_{j+1}$, then $s_i = K - j$. s_i is a number between 1 and K .

At the beginning, the whole table v is initialized to 0. When a packet arrives, the packet signature pkt and predecessor identifier p_{id} are extracted and hashed using each hash function h_i . s_i is written into entry $h_i(pkt, p_{id})$, if s_i is larger than the entry's current value. After current time t passes the required end time t_{end} , the optimal k_j is calculated among k_1, k_2, \dots, k_K which minimizes false positive rate f based on the actually inserted element number n using formula (1). The next job is to restore and archive the Bloom filter b with the optimal k_j hash functions. Each entry's value $v(i)$ is compared with $K - j$. If $v(i)$ is larger than $K - j$, the corresponding bit $b(i)$ in Bloom filter is set to 1.

- (1) Initialize the table v and Bloom filter b to 0
- (2) $n \leftarrow 0$
- (3) while ($t < t_{end}$ AND receive a new packet pkt with p_{id})
- (4) $n \leftarrow n + 1$;
- (5) for ($i \leftarrow 1$ to k_K)
- (6) $j \leftarrow h_i(pkt, p_{id})$
- (7) $v(j) \leftarrow \max(v(j), s_i)$
- (8) Find the index j of k_j which minimizes f in k_1, k_2, \dots, k_K
- (9) for ($i \leftarrow 1$ to m)
- (10) if ($v(i) > K - j$)
- (11) $b(i) \leftarrow 1$

Fig. 3. k -adaptive Procedure

We use the following example to demonstrate the k -adaptive mechanism in detail. Let $m = 10^6$, $Q = 2$ and $K = 3$. Suppose we want to choose the best k among $k_1 = 1$, $k_2 = 3$ and $k_3 = 4$, and the hash function sets are $H_1 = \{h_1\}$, $H_2 = \{h_1, h_2, h_3\}$, $H_3 = \{h_1, h_2, h_3, h_4\}$. Instead of recording 3 m -bit Bloom filters b_1 , b_2 and b_3 in the memory which use the hash set H_1 , H_2 , and H_3 respectively, we only keep a table v with m 2-bit entries. Now we show that we can restore b_1 , b_2 and b_3 using v . We observed that if finally an entry $b_1(i)$ is set to 1, then the entries $b_2(i)$ and $b_3(i)$

must also be 1, because $H_1 \subset H_2 \subset H_3$. Therefore, there are only 4 possible value combinations of the 3 entries $b_1(i)$, $b_2(i)$ and $b_3(i)$. As shown in Table I, we can find a bijection between $b_1(i)$, $b_2(i)$, $b_3(i)$ and the number of 1s in them, and thus we can use only 2 bits to represent the 3 entries. In this case, $s_1 = 3$, $s_2 = s_3 = 2$, and $s_4 = 1$, and they represent the number of 1s that the corresponding hash function will set in the 3 Bloom filters. If finally we receive $n = 2.5 * 10^5$ packets, we get the following false positive rates using formula (1): $f_{b_1} = 0.221$, $f_{b_2} = 0.147$, $f_{b_3} = 0.160$. Therefore, we choose $k = k_2 = 3$ and restore Bloom filter b_2 . We compare each entry $v(i)$ with $K - 2 = 1$ and set $b(i)$ to 1 if $v(i)$ is larger.⁴

TABLE I
BIJECTION BETWEEN $K = 3$ BLOOM FILTERS AND 2-BIT TABLE

$b_1(i)$	$b_2(i)$	$b_3(i)$	$v(i)$ (# of 1s)
0	0	0	0
0	0	1	1
0	1	1	2
1	1	1	3

We must point out that this k -adaptive mechanism can be used not only in Bloom filter-based IP traceback, but in other Bloom filter applications that the element number is not known a priori.

V. THEORETICAL ANALYSIS AND EXPERIMENTAL EVALUATION

In this section, we first analyze and evaluate the traceback performance of fully deployed TOPO, and then analyze TOPO when it is partially deployed. We focus on the performance comparison between TOPO and SPIE. We finally analyze the performance of our k -adaptive mechanism for Bloom filters.

A. Analysis under Full Deployment of TOPO

1) *Theoretical Analysis:* In the Bloom filter-based IP traceback systems, the traceback request is initiated by the victim (or IDS) when it detects intrusions. Finally the victim will get an attack graph from the IP traceback system which not only contains the real attack path, but also some extra innocent nodes because of the false positives of Bloom filters.

In SPIE's analysis [25], an upper bound of the expected number of extra nodes ex_all in the attack graph G is given by:⁵

$$E_{SPIE}(ex_all) = \frac{Ldf}{1 - df}, \quad (5)$$

where d is the maximum number of each router's predecessors on the network, and L is the number of routers on the attack path. This formula requires that $0 \leq df < 1$, otherwise it will not converge. However, this formula does not answer all the questions we exactly desire to know: How many unnecessary queries are sent out into the network? How many innocent end nodes are there in the attack graph? A good IP traceback system must have no or less unnecessary queries and innocent end nodes.

Let ex_query denote the total number of extra (unnecessary) queries that are sent out into the network, and ex_end denote the number of extra end nodes in the attack graph. We can easily get the following theorem for SPIE:

⁴When $K = 3$, we can effectively use logical operations instead of comparison operations to restore any desired Bloom filter from the table. b_1 can be restored by AND of the two bits in each entry of v ; b_2 is just the most significant bit of v ; b_3 can be restored by OR of the two bits in each entry of v .

⁵We get a slight different bound in Theorem 1.

Theorem 1:

$$E_{SPIE}(ex_query) = \frac{L(d-1)}{1-df}. \quad (6)$$

$$E_{SPIE}(ex_all) = f \frac{L(d-1)}{1-df} \quad (7)$$

$$= f \cdot E_{SPIE}(ex_query). \quad (8)$$

$$E_{SPIE}(ex_end) = (1-f)^d f \frac{L(d-1)}{1-df} \quad (9)$$

$$= (1-f)^d \cdot E_{SPIE}(ex_all). \quad (10)$$

Proof: Figure 4 demonstrates the tree structure of predecessors ($d = 3$ in this case) viewing along the reverse attack path: $Victim \rightarrow R_1 \rightarrow \dots \rightarrow R_L \rightarrow Attacker$. In real world it should be a merged net instead of a tree, because many routers share the same predecessors. Figure 4 shows the worst possible scenario, since we want to calculate the upper bound of the expectations. Each level consists of the innocent nodes which have the same probability to be queried. As shown in Figure 4, the number of nodes on level q is

$$L(d-1)d^{q-1}. \quad (11)$$

According to SPIE, the probability that an innocent node on level q is queried, the probability that it is falsely included in the attack graph, and the probability that it is end node in the attack graph are

$$f^{q-1}, \quad (12)$$

$$f^q, \quad (13)$$

and

$$(1-f)^d f^q \quad (14)$$

respectively. Therefore, we get equation (6), (7) and (9). ■

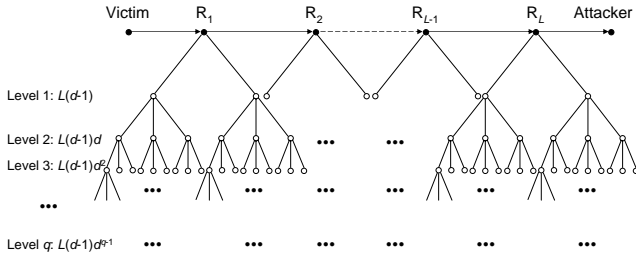


Fig. 4. Tree Structure of Predecessors

Now we analyze the traceback performance of TOPO. Let $F = 1 - (1-f)^d$. If $df \ll 1$, $F \approx df$. If we ignore the memory consumed by the predecessor list, we can get the following theorem.⁶

Theorem 2:

$$E_{TOPO}(ex_query) = f \cdot E_{SPIE}(ex_query). \quad (15)$$

$$E_{TOPO}(ex_all) = F \cdot E_{SPIE}(ex_all). \quad (16)$$

$$E_{TOPO}(ex_end) = (1-fF)^d E_{TOPO}(ex_all) \quad (17)$$

$$\approx F \cdot E_{SPIE}(ex_end). \quad (18)$$

Proof: In TOPO, the probability that an innocent node on level q is queried, the probability that it is falsely included in the attack graph, and the probability that it is end node in the attack graph are

$$f^q, \quad (19)$$

$$F f^q, \quad (20)$$

⁶Please refer to the discussion in Section VI-C to understand why the predecessor list memory size may be ignored.

and

$$(1-fF)^d F f^q \quad (21)$$

respectively. Therefore, we get equation (15), (16) and (17), and

$$E_{TOPO}(ex_end) = \frac{F(1-fF)^d E_{SPIE}(ex_end)}{(1-f)^d} \quad (22)$$

$$\approx F \cdot E_{SPIE}(ex_end).$$

As indicated in Theorem 2, the number of unnecessary queries in TOPO is only f times to that in SPIE, which is a significant improvement. Also, the numbers of extra nodes and extra end nodes reduce to F times to those in SPIE. If $f = 0.0001$, and $d = 100$, $F \approx 0.01$, which means that, compared with SPIE with the same resource, only 1% nodes will appear in the attack graph using TOPO. ■

2) *Experimental Study:* In our theoretical analysis, we have assumed that all routers have equal degree of predecessors. However, it is not realistic in real world networks, where the degrees of routers are different. In this experimental study, we use real world Internet topologies provided by CAIDA [4] to evaluate and compare the performance of SPIE and TOPO in traceback.

In our experiments, we use real world Internet topology captured on Nov. 5, 2005 from one of CAIDA's skitter monitor b-root.skitter.caida.org, which is a topology map viewed from a single origin (128.9.0.109) to 317, 218 destinations⁷. Each router is assumed to be equipped with Bloom filters that have the same false positive rate. We simulate the traceback from the single origin to every destination and calculate the expected number of extra queries, extra nodes and extra end nodes with respect to the false positive rate of Bloom filters. Figure 5 shows our experimental results.

As shown in Figure 5(a), as the false positive rate f of Bloom filters decreases, both SPIE and TOPO generate less extra queries. However, when f is less than 10^{-2} , the expected number of extra queries in SPIE almost remains the same (about 156), which indicates that there exists a lower bound in SPIE on expected number of extra queries. This can be explained using equation (6). Meanwhile, the expected number of extra queries in TOPO always decreases as f decreases. When $f = 0.01$, the expected number of extra queries is less than 2, while SPIE's expected number of extra queries is more than 165 for the same f . Figure 5(b) shows the expected number of extra nodes in the attack graph. As f decreases, both SPIE and TOPO create less extra nodes. However, TOPO has much smaller expected number of extra nodes than SPIE. Figure 5(c) shows that TOPO also has smaller expected number of extra end nodes compared to SPIE.

In sum, both the theoretical analysis and the experimental results show that TOPO has better traceback performance compared to SPIE. In other words, TOPO can achieve the same performance as SPIE with lower memory requirement on Bloom filters. For instance, TOPO with $f = 0.005$ can achieve better traceback performance than SPIE with $f = 0.0001$. This means that TOPO requires less memory allocated for Bloom filters on routers. Therefore, TOPO is more efficient with the same traceback capability.

B. Analysis under Partial Deployment of TOPO

It is difficult to analyze the performance of partially deployed Bloom filter-based IP traceback systems because of the varieties of deployment. To simplify the analysis, we consider partially deployed systems with the following constraint: On all possible paths in the partially deployed system, there is at least one Bloom filter-equipped router within any S steps.

⁷We only consider the destinations with completed paths in this data set.

VI. FURTHER DISCUSSIONS

In this section, we will discuss several considerations when designing and implementing TOPO.

A. Packet Signature Choice

Packet Signature can be flexible. There are different choices which can meet the requirement of distinguishing different packets. For instance, a subset of the IP header fields and the first several bytes of the packet payload are used in [20] and SPIE. PAS only uses a long excerpt of payload, which is useful when the exact packet header is unavailable. However, the excerpt must be long enough to identify different packets, and thus the attackers may avoid detect by attacking through a lot of packets with short payload. In TOPO, it is preferred to use a packet signature which contains IP header information.

B. Predecessor Identifier Choice

When a router has only one predecessor at each of its input port, for example, inner routers on the Internet, we choose the input port of each packet as its predecessor identifier. For a router that has multiple predecessors at one input port, we can use Layer 2 (i.e., data-link layer) information (such as source MAC addresses) to differentiate these multiple predecessors. For instance, on the Ethernet, the multiple predecessors and the router are connected through the broadcast-based transmission media. When the router receives a packet from one of its predecessors, the source MAC address of the header of the Ethernet frame can be used as the predecessor identifier. Similarly, ATM's VPI/VCI information can also be used for this purpose on ATM networks.

We propose to use local topology information in TOPO. The local topology information means the router's immediate predecessor and immediate successor. In the current system, we only use predecessor information. We believe the successor information can also be utilized. We will address this in the future research.

C. Predecessor List Memory Size

In most cases, the memory size of the predecessor list is much smaller than those of the Bloom filters, and that is why we ignore its influence on Bloom filters when we analyze the performance of TOPO in Section V. Generally, inner routers do not have a large number of predecessors (which are typically no more than 100), and these predecessors often remain unchanged for a long time. Therefore, we need not archive the predecessor list often. Instead, a router may maintain a static list to store all appeared predecessor identifiers for more than one Bloom filters, and only archive 1 bit for each predecessor on the list when archiving the Bloom filters: value 1 means that the router receives packets from that predecessor in that certain time interval, and value 0 not.

To find the direct support to our point of view from the real world Internet, we analyze CAIDA's Internet Topology Data Kit #0304 (ITDK0304) [5]. ITDK0304 is the skitter data of the Internet router-level graph collected between Apr 21 and May 8, 2003. There are a total of 192244 nodes and 636643 directed links. Table II shows the distribution of Internet routers' upstream degrees derived from ITDK0304. The average upstream degree is as low as 3.31, and the maximum upstream degree is only 269. Moreover, more than 99% routers have less than 25 upstream neighbors, and more than 99.98% routers have less than 100 upstream neighbors. These facts quite support our point of view that the memory sizes of predecessor lists need to be archived are adequately small compared with the large sizes of Bloom filters.

D. Membership Check Burden on Routers

In TOPO, suppose that there are d predecessors on one router's predecessor list. When the router receives a query message, it has to do d membership checks by combining the packet identifier with each predecessor identifier on the list, while the router in SPIE only need to do 1 membership check. It seems that TOPO complicates the membership check process, and thus aggravates the computation burden on each router. However, after using topology information in TOPO, the probability that each innocent router is queried becomes f times of that of SPIE. Let $member_check$ denote the number of membership checks, and we get that

$$E_{TOPO}(member_check) = df \cdot E_{SPIE}(member_check). \quad (32)$$

In most cases, $d \cdot f \ll 1$, therefore actually TOPO alleviates the membership check burden on each router which receives fewer query messages and does fewer membership checks.

E. Applying Compressed Bloom Filters

Mitzenmacher [21] introduced *compressed Bloom filters*. He proposed that Bloom filters can be compressed to improve their performance by achieving either a lower false positive rate with the same memory size, or smaller memory size with the same false positive rate. The compressed Bloom filters can be used to reduce the number of bits broadcast in sharing Web cache information. As shown in Table VI in [21], a compressed Bloom filter can achieve the same false positive rate as the standard Bloom filter while reducing the memory over 20%. The tradeoff costs are the increased processing requirement for compression and decompression and larger memory requirements at the endpoint machines.

In Bloom filter-based IP traceback systems, if a router stores a lot of archived pages of previous Bloom filters, and the received query messages are infrequent, the gain of applying compressed Bloom filters can overcome the processing costs introduced by compression and decompression. However, if a router only has a few archived Bloom filters, the memory overhead of implementing compression will be unacceptable. Furthermore, if the query messages are frequently received which always query different Bloom filters, the router would be busy in decompressing the required Bloom filters. The reason is that the router usually has no enough memory to keep two decompressed Bloom filters at the same time.⁹

F. Applying Hierarchical Bloom Filters

We also consider applying the Hierarchical Bloom filters [24] in TOPO. However, we find that actually hierarchical architecture has no benefit to false positive rate compared with the standard Bloom filters, and is even worse. The authors of [24] referred to the false positive rate of the standard Bloom filter upon which their Hierarchical Bloom filters are built as basic false positive rate f_o , and referred to the false positive rate resulting from their Hierarchical Bloom filters as effective false positive rate f_e . They showed that $f_e \ll f_o$. We agree with it. However, hierarchy also introduces more inserted elements into the Bloom filters, which increases the false positive rate and achieve no benefits eventually. Suppose n packets are inserted into a Hierarchical Bloom filter using q different strings for each, and the totally inserted elements are $n_0 = qn$. Then we get

$$f_e = f_o^q \approx (0.6185^{\frac{m}{n_0}})^q = 0.6185^{\frac{m}{n}} = f, \quad (33)$$

⁹Usually, the size of decompressed Bloom filters is over 10 times larger than that of the original Bloom filters.

TABLE II
DISTRIBUTION OF INTERNET ROUTERS' UPSTREAM DEGREES

Upstream Degree	0 - 24	25 - 49	50 - 74	75 - 99	100 - 124	125 - 274	≥ 275	Average: 3.31
Number of Routers	190469	1501	191	52	20	11	0	Total: 192244
Percent of Routers	99.0767%	0.7808%	0.0994%	0.0270%	0.0104%	0.0057%	0	

where f is exactly the false positive rate when the standard Bloom filter inserts n packets each using just one string. Therefore there is no benefit to false positive rate using hierarchy. It makes false positive rate even worse considering that hierarchy needs to check all possible alignments of payload excerpt.

VII. CONCLUSION

Several Bloom filter-based IP traceback schemes have been proposed. However, Bloom filters' inherent false positives restrain the effectiveness of previous schemes. In this paper, we have proposed TOPO, a topology-aware single packet IP traceback system, in which the predecessor information is used for traceback purpose. Our analysis showed that TOPO significantly reduces not only the number of unnecessary queries but also the false attributions. In addition, practicability is an important and desired property of IP traceback systems. We have studied the partial deployment problem of Bloom filter-based IP traceback systems and carefully designed to allow TOPO to be partially deployed while maintaining its traceback capability. We also proposed k -adaptive mechanism for Bloom filters which control parameters may be adaptively adjusted according to the number of actual received packets. Such adjustments can help Bloom filters-based IP traceback systems to achieve the best performance in terms of false positive rate and storage space requirement when the number of arrival packets varies significantly over time.

In the future, we will continue to improve the design of TOPO in terms of processing overhead and memory space requirement.

ACKNOWLEDGMENTS

This work was partially sponsored by ARDA under contract number NBCHC030107, NSF under contract number 0313837, and Carver Trust Foundation. We thank the anonymous reviewers for valuable comments.

REFERENCES

- [1] LAND Attack. [Online]. Available: <http://www.insecure.org/splotts/land.ip.DOS.html>
- [2] Ping of Death Attack. [Online]. Available: <http://www.insecure.org/splotts/ping-o-death.html>
- [3] Teardrop Attack. [Online]. Available: <http://support.microsoft.com/kb/q179129/>
- [4] "CAIDA's skitter project." [Online]. Available: <http://www.caida.org/tools/measurement/skitter/>
- [5] "CAIDA's Internet Topology Data Kit #0304," Cooperative Association for Internet Data Analysis, San Diego Supercomputer Center (SDSC), University of California, San Diego (UCSD), 2003. [Online]. Available: http://www.caida.org/tools/measurement/skitter/router_topology/
- [6] A. Belenky and N. Ansari, "IP traceback with deterministic packet marking," *IEEE Communications Letters*, vol. 7, no. 4, pp. 162–164, Apr. 2003.
- [7] S. M. Bellovin, "ICMP traceback messages," Internet Draft, 2000.
- [8] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, July 1970.
- [9] A. Broder and M. Mitzenmacher, "Network applications of Bloom filters: A survey," in *Proceedings of the 40th Annual Allerton Conference on Communication, Control, and Computing*, Monticello, USA, Oct. 2002, pp. 636–646.

- [10] H. Burch and B. Cheswick, "Tracing anonymous packets to their approximate source," in *Proceedings of USENIX LISA 2000*, New Orleans, USA, Dec. 2000, pp. 319–327.
- [11] B. Cheswick, H. Burch, and S. Branigan, "Mapping and visualizing the Internet," in *Proceedings of 2000 USENIX Annual Technical Conference*, San Diego, USA, June 2000.
- [12] D. Dean, M. Franklin, and A. Stubblefield, "An algebraic approach to IP traceback," *Information and System Security*, vol. 5, no. 2, pp. 119–137, 2002.
- [13] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, "Summary cache: a scalable wide-area Web cache sharing protocol," *IEEE/ACM Transactions on Networking*, vol. 8, no. 3, pp. 281–293, 2000.
- [14] L. A. Gordon, M. P. Loeb, W. Lucyshyn, and R. Richardson, "2005 CSI/FBI computer crime and security survey," 2005. [Online]. Available: <http://www.usdoj.gov/criminal/cybercrime/CSI.FBI.htm>
- [15] R. Govindan and H. Tangmunarunkit, "Heuristics for Internet map discovery," in *Proceedings of IEEE INFOCOM 2000*, Tel Aviv, Israel, Mar. 2000, pp. 1371–1380.
- [16] B. Huffaker, D. Plummer, D. Moore, and k claffy, "Topology discovery by active probing," in *Proceedings of the 2002 Symposium on Applications and the Internet (SAINT 2002)*, Nara, Japan, Jan. 2002.
- [17] J. Li, M. Sung, J. Xu, and L. Li, "Large-scale IP traceback in high-speed internet: Practical techniques and theoretical foundation," in *Proceedings of 2004 IEEE Symposium on Security and Privacy*, Oakland, USA, May 2004.
- [18] C. Lynn, W. Milliken, and W. T. Strayer, "SPIE memory requirements reduction," BBN Technologies, Tech. Rep. BBN REPORT-8385, Dec. 2003.
- [19] A. Mankin, D. Massey, C.-L. Wu, S. F. Wu, and L. Zhang, "On design and evaluation of "Intention-Driven" ICMP traceback," in *Proceedings of 10th IEEE International Conference on Computer Communications and Networks*, Scotsdale, USA, Oct. 2001.
- [20] S. Matsuda, T. Baba, A. Hayakawa, and T. Nakamura, "Design and implementation of unauthorized access tracing system," in *Proceedings of the 2002 Symposium on Applications and the Internet (SAINT 2002)*, Nara, Japan, Jan. 2002.
- [21] M. Mitzenmacher, "Compressed Bloom filters," *IEEE/ACM Transactions on Networking*, vol. 10, no. 5, pp. 604–612, Oct. 2002.
- [22] K. Park and H. Lee, "On the effectiveness of probabilistic packet marking for IP traceback under denial of service attack," in *Proceedings of IEEE INFOCOM 2001*, Anchorage, USA, Apr. 2001, pp. 338–347.
- [23] 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, June 2001.
- [24] K. Shanmugasundaram, H. Brönnimann, and N. Memon, "Payload attribution via hierarchical Bloom filters," in *Proceedings of the 11th ACM Conference on Computer and Communications Security*, Washington DC, USA, Oct. 2004.
- [25] 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.
- [26] D. Song and A. Perrig, "Advanced and authenticated marking schemes for IP traceback," in *Proceedings of IEEE INFOCOM 2001*, Anchorage, USA, Apr. 2001.
- [27] E. H. Spafford, "OPUS: Preventing weak password choices," *Computers & Security*, vol. 11, no. 3, pp. 273–278, May 1992.
- [28] R. Stone, "Centertrack: An IP overlay network for tracking DoS floods," in *Proceedings of the 9th USENIX Security Symposium*, Denver, USA, Aug. 2000, pp. 199–212.
- [29] W. T. Strayer, C. E. Jones, F. Tchakountio, and R. R. Hain, "SPIE-IPv6: Single IPv6 packet traceback," in *Proceedings of the 29th IEEE Local Computer Networks Conference (LCN)*, Tampa, USA, Nov. 2004.
- [30] S. F. Wu, L. Zhang, D. Massey, and A. Mankin, "Intention-Driven ICMP trace-back," Internet Draft, 2001.