Tradeoffs in Probabilistic Packet Marking for IP Traceback.

Micah Adler
Department of Computer Science
University of Massachusetts, Amherst

Abstract

Recently, a number of probabilistic packet marking schemes have been proposed for the problem of tracing a sequence of packets back to an anonymous source. This paper introduces a new technique for marking that applies when the packets are all sent along the same path. This new technique allows this path to be traced even when there is only a single bit in the header allocated to the marking scheme. If the path is encoded using n bits, then with this scheme, for any constant $\epsilon > 0$, the number of packets required to reconstruct the path is $O((2+\epsilon)^{2n})$. We also demonstrate that $\Omega(2^n)$ packets are necessary if only one bit is used. To contrast our new technique with existing protocols, we demonstrate that all existing protocols belong to a class of algorithms for which at least $\log n$ bits are required. We also study the tradeoff between b, the number of header bits used, and the number of packets required. We provide a protocol such that $O(bn^2 2^b (2+\epsilon)^{4n/2^b})$ packets are sufficient, for any constant $\epsilon > 0$. This protocol is simple enough to be quite effective in practice. We also provide an information theoretic lower bound demonstrating that $\Omega(2^b 2^{n/2^b})$ packets are necessary. These are the first results concerning optimal tradeoffs in probabilistic packet marking.

1 Introduction

In recent years, the Internet has seen an alarming increase in what are known as denial-of-service attacks. Such an attack consists of a malicious party sending huge volumes of traffic to a remote host or a network, thereby denying legitimate access to these resources. Unfortunately, such attacks are easy to perform, and in fact there are well known techniques for mounting attacks that are coordinated between a large number of distributed hosts [3]. To make matters worse, in the current and foreseeable routing architectures of the Internet, a host transmitting packets can use a forged (or *spoofed*) source address for those packets. This means that there is little or no accountability for the source of these attacks. Thus, one of the most important tools that is needed to help fight denial-of-service attacks is the ability to trace packets back to their source. This is known as the *IP Traceback* problem.

A number of different approaches to the IP Traceback problem have been suggested. For example, *Ingress Filtering* [5] requires network providers to prevent spoofing by blocking a packet at or near its source if it is sent with an incorrect source address. Network providers have resisted Ingress Filtering for a variety of reasons, including the resulting degradation in performance, the additional administrative overhead involved, and the reliance on spoofing by some of the techniques proposed for mobile networking [10]. Furthermore, Ingress Filtering is only effective as a means of enforcing accountability in the Internet if performed universally, or nearly universally. Other approaches to the IP Traceback problem include requiring routers to keep a log of forwarded packets [13], and

generating additional traceback packets that routers send to the destination of a regular packet [1]. Shortcomings of these and other techniques are discussed in [2] and [12].

Recently, [12] proposed the following clever approach to the IP Traceback problem. Some fixed number of bits in the packet header are allocated to IP Traceback. Those bits are used to store a router ID, and a hop count. Every router that forwards a packet, independently with some probability p, writes its unique ID to those bits, and sets the hop count to 0. With probability $1-p$, the router ID is left unchanged, and the hop count is incremented. Now, say an *attacker* is performing a denial-of-service attack on a *victim* by sending a stream of packet along a path of length ℓ. If $p \approx 1/\ell$, then after the victim has received $O(\ell \log \ell)$ packets, with high probability this scheme provides the victim with the entire path back to the attacker.

Note that since some packets may not be marked by any intervening router, the attacker might be able to insert fake information, possibly making the path look like it extends beyond its actual source. Various ways of dealing with this have been suggested, including using cryptographic techniques [14], or knowledge of the routing topology [12]. For simplicity, we assume here that it is sufficient to determine a path that contains the correct path to the attacker as a suffix.

It turns out that the number of header bits required by this scheme is too large to make it practical (header bits are a quite valuable resource). Thus, [12] also suggests the following: instead of sending the ID of each router in one piece, partition the ID into t pieces, and send a randomly chosen piece along with a $\log t$ bit description of which piece is being sent. This reduces the number of header bits required, but increases the number of packets required to $O(\ell k \log(\ell k))$. They optimize this scheme to one that requires 16 header bits, and can reconstruct the entire path with high probability after a few thousand packets have been received.

The scheme proposed in [12] has produced a flurry of activity in the networking community. This kind of technique is now referred to as *probabilistic packet marking*, or PPM. In a relatively short time, a number of variations on PPM have been introduced to improve and further analyze the [12] technique [4, 8, 14] (see also [7] and [9]). One important concern is reducing the number of header bits required for PPM. Prior to this paper, the minimum known number of required bits was 13, achieved by a scheme in [4]. Also, some of the approaches are computationally expensive. Despite the number of papers in this area, a rigorous theoretical analysis of PPM has been lacking: There is no real understanding of the interplay and inherent tradeoffs between the number of header bits used and the number of packets required to reconstruct (with high probability) the path used by the attacker.

Let b be the number of header bits allocated to IP Traceback, and let n be the number of bits required to represent a path of attack. The previous work in this area does not study the relationship between these parameters, but all previous PPM protocols belong to a class of protocols for which b must be at least $\log n$ for the victim to be able to determine the n-bit path description with probability greater than $1/2$. In particular, all previous protocols encode a path ρ using a subset $S(\rho)$ of the set of 2^b possible b-bit headers. The subset $S(\rho)$ is unique for each possible path of attack. All packets sent along ρ arrive at the victim with the b bits set to one of the elements of $S(\rho)$, and if enough packets are sent along ρ, then, with high probability, the victim receives at least one of each element of $S(\rho)$. This tells the victim the set $S(\rho)$, and thus the path ρ as well. For example, the scheme described above from [12] has this property.

To see the lower bound on this class of protocols, note that there are 2^{2^b} different subsets S that are possible. Since there are 2^n possible n-bit strings, if the victim must determine the correct path

with probability at least 1/2, then 2^{2^b} must be at least $2^n/2$. Assuming that $n \geq 2$, the fact that b must be an integer implies that $b \geq \log n$.

In this paper, we introduce a new PPM technique that allows for a much more efficient encoding of the path description. This new kind of encoding takes advantage of the distribution of different packet headers received at the victim. In other words, to determine the path of attack, the victim examines *how many* of each possible header it has received, and not just what subset of the possible headers is received. This new encoding technique can be implemented with the same restrictions as the previous protocols: the routers along the path of attack operate in a completely distributed fashion, and the protocol does not require any state information at a router. Due to the memoryless nature of Internet routing, the lack of state information is an important requirement for PPM protocols: it is impractical for routers to store any information on individual flows. The computation required by this protocol at the routers is extremely fast and simple, and the computation required at the victim is also efficient.

As in [8] and [12], we here consider the important special case of the problem where the attacker sends all of its packets along the same path. With this assumption, our new technique allows any path description to be revealed to the victim even when $b = 1$. In other words, this new scheme requires using only a single header bit, which is obviously the minimum possible. Unfortunately, this requires $\Theta((2+\epsilon)^{2n})$ packets to be received by the victim, for any constant $\epsilon > 0$, and thus is only appropriate for small values of n. However, we also provide an information theoretic lower bound demonstrating that $\Omega(2^n)$ packets are necessary for any protocol where the victim is able to determine the correct path with probability greater than 1/2.

In this paper we also study the case where $b > 1$. In fact, we are able to obtain a good understanding of the optimal tradeoff between the number of header bits used and the number of packets required in the case where there is only a single path used by the attacker. We provide a protocol, as well as an information theoretic lower bound, demonstrating that the optimal number of packets that must be received for given values of n and b grows exponentially with n, but decreases *doubly* exponentially with b. Specifically, $O(bn^2 2^b (2+\epsilon)^{4n/2^b})$ packets are sufficient, for any constant $\epsilon > 0$, and $\Omega(2^b 2^{n/2^b})$ packets are necessary. These are the first results concerning optimal tradeoffs in probabilistic packet marking.

2 The Model

In this paper, we use two slightly different models for the upper bound and for the lower bound. The lower bound model is strictly more powerful than the upper bound model: lower bounds for that model imply lower bounds for the upper model, and upper bounds for the upper bound model imply upper bounds for the lower bound model. The actual scenario of interest lies somewhere between the two models; we address this in more detail in Section 5.

For the upper bound model, there is a network $\mathcal{N}$ consisting of a rooted tree. The root of the tree is the victim. At the start of the attack, the attacker chooses a node of the tree, and then sends packets to the victim along the path in the network from the chosen node to the victim. We here assume that the attacker only chooses a single location in the network: there is only one path of attack. The case where the attacker is able to use multiple paths is an interesting open problem.

For ease of exposition, we introduce the protocols by first making a number of simplifying assump-

tions about the network. We demonstrate in Section 5 that our results can easily be extended to hold in scenarios where these assumptions are relaxed. In particular, we first assume that the tree is a complete binary tree. We also assume that when a packet is sent to a node, that node is able to distinguish which child of the node the packet came from. Finally, we also assume that the victim has complete knowledge of the network topology. Again, we demonstrate in Section 5 that each of these assumptions can be removed.

The header of each packet contains b bits that are allocated to traceback information. No other bits of the packet can be utilized, and thus we shall simply assume that each packet consists of only these b bits. For each packet that is forwarded from the attacker to the victim, each of the intermediate nodes is allowed to alter these b bits. However, the intermediate nodes are memoryless, and thus, for each node, the new set of b bits that a node forwards to its parent in the tree can only be a function of the incoming b bits, which child of that node the packet arrives from, and some random bits. For the upper bound, we also assume that the nodes do not know their location in the network, and thus every node is required to use the same algorithm. The victim, on the other hand, does have storage. The objective is for the intermediate nodes to forward packets to the victim in such a manner that after collecting enough of these packets, the victim is able to determine a simple path in $\mathcal{N}$ from itself to another node such that the attacker is one of the nodes on this path.

For a given placement of the adversary at a distance $n+1$ from the root, we shall refer to the node on the path from the root to the attacker at distance i from the root as N_i (where the victim is N_0, and the attacker is N_{n+1}). The victim directly sees which child it receives packets from. Since we are assuming a binary tree, we can represent the remainder of the path as a binary string $B = B_1 B_2 \dots B_n$, where $B_i = 0$ if the path goes to the left child of N_i, and $B_i = 1$ otherwise. Note that when determining the outgoing bits for any packet, node N_i has local access to one bit of the string B: specifically the bit B_i.

We refer to the entire sequence of messages sent from the adversary to the victim as the transmission of a single packet (where the bits contained in that packet may change as it makes its way through the network). We are concerned with the tradeoff between b and the number of packets that must be sent in order for the victim to know the string B with probability at least $1 - \Delta$.

For the lower bound model, we assume that there are three nodes in the system: the adversary, an intermediate node, and the victim. The intermediate node has an n-bit string to send to the victim. The adversary is allowed to send any b-bit packet to the intermediate node. For each such packet, the intermediate node forwards that packet on to the victim. The bits sent by the intermediate node are allowed to depend on the bits sent to it by the adversary, as well as the n-bit string it holds, as well as an unlimited number of random bits, but nothing else. In particular, the intermediate node has no memory, and cannot remember previous packets it has sent, not can it remember previous used random bits. Clearly any algorithm for the upper bound model can be simulated in the lower bound model, and thus the lower bound model is strictly more powerful than the upper bound model.

3 The upper bound

We start by providing an algorithm for the case where $b = 1$. The basic idea behind the protocol is to encode the string B into p, the probability that the bit received by the victim is a 1. For

example, consider the encoding where $p = \sum_{i=1}^{n} B_i(\frac{1}{2})^i$. With this encoding, we show below that it is sufficient that the victim obtains enough samples to determine the bias of p (with the required confidence) within an additive term of $(\frac{1}{2})^{n+2}$. All of our protocols use this general kind of an encoding to transmit bits to the victim, and so the following lemma is a useful tool for analyzing this kind of protocol.

Lemma 1 *Consider any set of bits $B_1 \ldots B_\ell$, and any protocol where the victim is able to determine real numbers p, σ, and $c_1 \ldots c_\ell$, that satisfy the following conditions:*

1. $|p - \sum_{j=1}^{\ell} c_j B_j| \leq \sigma$.
2. *For all i, $1 \leq i \leq \ell - 1$, $c_i > 2\sigma + \sum_{j=i+1}^{\ell} c_i$.*
3. $c_\ell > 2\sigma$.

These values uniquely determine the bits $B_1 \ldots B_\ell$.

Proof: We demonstrate that for any i, $1 \leq i \leq \ell$, if the victim knows $B_1 \ldots B_{i-1}$, then it can determine the value of B_i. To do so, let $p' = p - \sum_{j=1}^{i-1} c_j B_j$. If $p' \geq c_i - \sigma$, then it must be the case that $B_i = 1$. On the other hand, if $p' < c_i - \sigma$, it must be the case that $B_i = 0$. Thus, the values of the B_i can be computed in a greedy fashion, starting with B_1 and working one bit at a time towards B_ℓ. ■

Let **DECODE**$(p, \sigma, c_1, \ldots, c_\ell)$ be the result of performing this decoding using the real numbers p, σ, and $c_1, \ldots, c_\ell$. It turns out that the encoding (described above) where $c_i = (\frac{1}{2})^i$ is not sufficient for our purposes. Instead, we use $c_i = \frac{r^{i-1}}{2}$, for $r = 1/2 - \epsilon$, for any ϵ such that $0 < \epsilon < 1/2$. To describe the protocol that actually achieves such an encoding, we describe what any node N_i does on the receipt of a bit from its neighbor. Note that there are only four possible inputs for the node N_i, differentiated by the bit B_i and the bit that N_i receives from node N_{i+1}. The following table describes the probability that N_i forwards a 1 to node N_{i-1} on the four possible inputs:

	Incoming bit	
B_i	0	1
0	0	$1/2 - \epsilon$
1	1/2	$1 - \epsilon$

Otherwise, in all four cases, N_i forwards a 0.

The victim now uses the following decoding algorithm: Obtain $t = O\left((1/\epsilon r^n)^2 \log(1/\Delta)\right)$ samples. Let x be the number of 1s in this set of samples, and let $p = x/t - r^n/2$. Let $\sigma = r^n/2 + \epsilon r^n$. Set the bits $B_1 \ldots B_n$ according to the process **DECODE**$(p, \sigma, \frac{1}{2}, \frac{r}{2}, \frac{r^2}{2}, \ldots, \frac{r^{n-1}}{2})$.

Theorem 1 *With probability $1 - \Delta$, this process returns the correct answer.*

Proof: Let p_i^0 (p_i^1) be the probability that the bit received by node N_i is a 1, given that the adversary starts with a 0 (1, respectively).

Claim 1 *For $r = 1/2 - \epsilon$, we have that $p_0^0 = \sum_{i=1}^{n} B_i \frac{r^{i-1}}{2}$, and $p_0^1 = r^n + \sum_{i=1}^{n} B_i \frac{r^{i-1}}{2}$.*

Proof: We see that for $i \leq n$, if $B_i = 0$, then $p_{i-1}^e = rp_i^e$, for $e \in \{0,1\}$. If $B_i = 1$, then $p_{i-1}^e = (1-\epsilon)p_i^e + \frac{1}{2}(1-p_i^e) = rp_i^e + \frac{1}{2}$, for $e \in \{0,1\}$. The claim now follows by induction. ■

From this claim, we see by a Chernoff bound that $\Pr[|p - p_0^0| > r^n/2 + \epsilon r^n] \leq \Delta$. When $|p - p_0^0| \leq r^n/2 + \epsilon r^n$, condition 1 of Lemma 1 is satisfied. To see that condition 3 is always satisfied, note that $r^n/2 + \epsilon r^n < r^{n-1}/4$ is equivalent to requiring that $r(\frac{1}{2} + \epsilon) < 1/4$, which follows from the fact that $r = 1/2 - \epsilon$. To see that condition 2 is also always satisfied, note that for all i, $1 \leq i \leq \ell - 1$, $c_i - \sum_{j=i+1}^{\ell} c_i > c_n$. Thus, by Lemma 1, the victim is able to determine the entire string with probability $1 - \Delta$. ■

Note that this algorithm leads to an exponential dependence on n. We show in Section 4 that for the case where $b = 1$, such a dependence is necessary. Since this makes the protocol impractical for all but small values of n, we also consider larger values of b. We next describe how to extend the one bit scheme to the case where $b > 1$. One approach would be to run b versions of the one bit scheme in parallel, using a single bit and disjoint sets of nodes for each scheme. For example we could partition the n nodes into b sets of size n/b (for now, ignoring rounding), where each set consists of the nodes N_i for $i = k \bmod b$, for $0 \leq k \leq b-1$. This approach does in fact lead to a valid protocol. Since the effective value of n decreases linearly with b, this leads to an exponential decrease in the number of samples required as b increases.

However, it turns out that this is far from optimal. In fact, it is possible to obtain a doubly exponential decrease in the number of samples required as b increases. To do so, instead of partitioning the nodes into b sets, we partition them into $d = 2^{b-1}$ sets, numbered 0 to $d-1$. Node N_i is in the set $i \bmod d$. Each of these sets will perform the one bit protocol (almost) independently. To see how to develop this idea into a valid protocol, we first consider an idealized scenario, where for every packet, the adversary sets the b bits by choosing randomly from the uniform distribution over all 2^b possible settings. We then describe how to convert this into a protocol where the adversary can set the b initial bits arbitrarily.

Denote the b bit positions as $d_0 \ldots d_{b-1}$. For the idealized setting, in each packet, the nodes in one cell of the partition perform the one bit protocol using the bit d_0. The remainder of the bits are used as a counter to specify which cell of the partition participates in the one bit protocol. In particular, for a particular packet P, let I_i^P be the integer corresponding to the binary representation of the bits $d_1 \ldots d_{b-1}$ that are received at node N_i. Thus, we say node i sets $I_{i-1}^P = j$ to mean that on packet P, the bits $d_1 \ldots d_{b-1}$ sent from node N_i to node N_{i-1} are set to the binary representation of j. Each node N_i performs the following protocol for each packet P:

- If $I_i^P = 0$, then perform the one bit protocol using bit d_0. Forward d_0 as the resulting bit of the one bit protocol, and set $I_{i-1}^P = 1$.
- Otherwise, forward d_0 unchanged, and set $I_{i-1}^P = I_i^P + 1 \bmod d$.

It is not hard to see that each cell k of the partition performs the one bit protocol on a subset of the bits. Let t_k be the number of samples received that perform the one bit protocol for cell k of the partition. Let $d = 2^{b-1}$. If the total number of samples received is $\Theta\left(d(1/\epsilon r^{n/d})^2 \log(d/\Delta)\right)$, then as long as $n \gg d$, $t_k \geq \Theta\left((1/\epsilon r^{n/d})^2 \log(d/\Delta)\right)$ with probability at least $1 - \Delta/2d$. This result

follows from a Chernoff bound; we omit the details since it is subsumed by the analysis of the full version of the protocol below. For this value of t_k, the effect is identical to performing the one bit protocol on a set of n/d bits. Thus, the victim is able to reconstruct all of the bits in cell k of the partition with probability $1 - \Delta/2d$. Thus, the victim is able to reconstruct all of the bits with probability $1 - \Delta$.

To make this algorithm work for an adversary that is allowed to set the initial bits arbitrarily, we need to modify the protocol slightly. Note that otherwise the adversary could, for example, set the initial bits to the same value for every packet, which would only inform the victim of the bits in one cell of the partition. The change to the protocol is simple: with a probability ρ (which can be any probability such that $0 \leq \rho \leq 1$, but ideally $\rho = 1/n$), each node N_i performs what is called a *reset*: it ignores the incoming bits completely, and sets $I^P_{i-1} = 1$. Bit d_0 is forwarded as B_i with probability $\frac{1}{2}$ and as 0 otherwise. This has the effect of resetting the counter with some probability, thereby allowing the bits from every partition to be sent to the victim.

With this more powerful adversary, we also need to develop a more complicated decoding procedure. The proof that this procedure is able to successfully decode is somewhat involved. We start with some intuition for this decoding algorithm. Let r^n_k be the probability that a packet P is reset by some node between N_n and N_1, inclusive, and $I^P_0 = k$. If $z(n,k)$ is the number of integers i, $1 \leq i \leq n$, such that $i \bmod d = k$, we see that $r^n_k = \sum_{j=1}^{z(n,k)} \rho(1-\rho)^{(j-1)d+k-1}$. We shall later use the fact that if $\rho \leq 1/n$, then $r^n_k \geq \frac{z(n,k)\rho}{e}$. Also, let α^k_j be the probability that a packet that arrives at N_0 is reset last by some node between N_n and $N_{k+(j-1)d}$, given that it is reset by some node between N_n and N_1, and that $I^P_0 = k$. From Bayes rule, we obtain $\alpha^k_j = \frac{1}{r^n_k} \sum_{t=j}^{z(n,k)} \rho(1-\rho)^{(j-1)d+k-1}$.

Let P_k be the set of packets P such that $I^P_0 = k$. For $0 \leq k \leq d-1$, let q^n_k be the fraction of packets in P_k such that no node between N_1 and N_n (inclusive) performs a reset on the packet. Note that q^n_k is not a value readily available to the victim; an important portion of the decoding algorithm is computing for each k a value $\bar{q}^n_k$ that serves as an estimate for q^n_k. Consider a packet chosen uniformly at random from the set of packets in P_k for which the adversary sets $d_0 = t$, for $t \in \{0,1\}$. The probability that the packet has d_0 set to 1 when it arrives at N_0 is

$$p^t_k = t \cdot q^n_k r^{z(n,k)} + \sum_{j=1}^{z(n,k)} B_{k+(j-1)d}(q^n_k + (1-q^n_k)\alpha^k_j)\frac{r^{j-1}}{2}.$$

Thus, if we knew exactly the values q^n_k, the decoding process would not be very different from the single bit protocol. However, without at least a fairly accurate estimate for q^n_k, such a decoding process would not have enough information. We next describe a decoding algorithm that computes such an estimate. We here describe this algorithm for the case where the value of n is known. However, the same process applies for any value of $\ell \leq n$ determined by the victim: the victim can decode any prefix of the path up to the adversary. We here also describe the easier case of the decoding process where $\rho \leq \frac{1}{n}$. We also want to emphasize that for this version of the paper, we have made no attempt to optimize the constants of the algorithm. We strongly suspect that the constants appearing here can be significantly improved; we shall address this in a later version. The algorithm works as follows:

- N_0 waits until it has received $F = \left(\left(\frac{16e^2}{\rho r^{\lfloor n/d \rfloor}}\right)^2 \left(\frac{4ed}{n\rho}\right) \ln(4d/\Delta)\right)$ packets.

- For $0 \leq k \leq d-1$, $j \in \{0,1\}$, let f_k^j be the total number of packets in P_k for which the value of the bit d_0 received at N_0 is j.
- Let $\bar{q}_k^n = \frac{f_k^1 + f_k^0 - r_k^n \cdot F}{f_k^1 + f_k^0}$.
- For $k = 0$ to $d-1$:
  - For $j = 1$ to $z(n,k)$,
    - Let $c_j^k = (\bar{q}_k^n + (1-\bar{q}_k^n)\alpha_j^k)\frac{r^{j-1}}{2}$.
  - Let $\sigma_k = (\bar{q}_k^n + (1-\bar{q}_k^n)\alpha_{z(n,k)}^k)\frac{r^{z(n,k)}}{2}$.
  - Let $p_k = \frac{f_k^1}{f_k^1+f_k^0} - \frac{\bar{q}_k^n r^{z(n,k)}}{2}$.
  - Set the bits B_t, for $t = k + (j-1)d$, $1 \leq j \leq z(n,k)$, according to the process **DECODE**$(p_k, \sigma_k, c_1^k, \ldots, c_{z(n,k)}^k)$.

Note that in the case that $\rho = \Omega(1/n)$, the number of packets required by this algorithm is $O\left(\frac{2^b n^2}{r^{4n/2^b}} \ln(2^b/\Delta)\right)$. Also note that the interesting case of the algorithm is when $2 \leq b \leq \lceil \log n \rceil$, since the first algorithm described above handles the case when $b = 1$, and when $b > \lceil \log n \rceil$, then techniques such as those used in [12] are sufficient.

Theorem 2 *If $2 \leq b \leq \lceil \log n \rceil$, then with probability at least $1 - \Delta$, this process returns the correct values of B_i, $\forall i$, $1 \leq i \leq n$.*

Proof: We here show that each of the d decoding processes produces the correct answer with probability at least $1 - \Delta/d$, from which the theorem follows directly from a union bound. For each call to the decode process, we demonstrate that the conditions of Lemma 1 are satisfied. For condition 2, we must show that for $1 \leq j \leq z(n,k) - 1$, $c_j^k > 2\sigma_k + \sum_{t=j+1}^{z(n,k)} c_t^k$. Note that since the expression $(\bar{q}_k^n + (1-\bar{q}_k^n)\alpha_j^k)$ is monotonically nonincreasing as j increases, and $r < 1/2$, we see that $c_j^k - \sum_{t=j+1}^{z(n,k)} c_t^k > r^{z(n,k)-1}/2$. Since $\bar{q}_k^n + (1-\bar{q}_k^n)\alpha_{z(n,k)}^k \leq 1$, we have that $\sigma_k \leq r^{z(n,k)-1}/4$, implying condition 2. Also note that condition 1, i.e., that $c_{z(n,k)}^k > 2\sigma_k$, follows directly from the definitions of $c_{z(n,k)}^k$ and σ_k, and the fact that $r < 1/2$.

Thus, we only have left to prove that condition 1 holds with probability at least $1 - \Delta/d$, or that $\Pr\left[\left|p_k - \sum_{j=1}^{z(n,k)} c_j^k B_{(j-1)d+k}\right| \leq \sigma_k\right] \geq 1 - \Delta/d$. Recall that p_k^t is the probability that a packet chosen uniformly at random the set of packets in P_k that start with $d_0 = t$, for $t \in \{0,1\}$, arrives at N_0 with d_0 set to 1; an expression for p_k^t is given above. If it were the case that (a) $q_k^n = \bar{q}_k^n$, (i.e., if our estimates of q_k^n were exact), and (b) $p_k^0(f_k^1 + f_k^0) \leq f_k^1 \leq p_k^1(f_k^1 + f_k^0)$ (which occurs if the fraction of packets for which $d_0 = 1$ at N_0 is exactly the expectation), then a value of $\sigma_k = q_k^n r^{z(n,k)}/2$ would be sufficient. Of course, the probability of these random variables being exactly their expectation is too small for our purposes. However, we can demonstrate that, with sufficiently high probability, they do not deviate far from their expectation.

To do so, we use two versions of the Chernoff bound [11]. In particular, if $X_1 \ldots X_t$ are i.i.d. random variables, such that $\Pr[X_i = 1] = p$, and $\Pr[X_i = 0] = 1 - p$, then for any δ such that $0 \leq \delta \leq 1$,

$$\Pr\left[\sum_{i=1}^{t} X_i \geq (1+\delta)tp\right] \leq e^{-\delta^2 tp/3},$$

and

$$\Pr\left[\sum_{i=1}^{t} X_i \leq (1-\delta)tp\right] \leq e^{-\delta^2 tp/2}.$$

We first use these bounds to demonstrate that it is likely that P_k is large enough to provide good estimates on the quantities of interest. In particular, we show the following:

Claim 2 *Let* $\mu = \left(\frac{16e^2}{\rho r^{z(n,k)}}\right)^2 \ln(4d/\Delta)$. $\Pr\left[f_k^0 + f_k^1 < \mu\right] \leq \frac{\Delta}{4d}$.

Proof: Regardless of what the adversary does, for any packet P, $\Pr[I_0^P = k] \geq r_k^n$. Thus, we can define a set of i.i.d. indicator variables $X_1 \ldots X_T$, where T is the total number of packets sent, such that $X_j = 1$ if packet j is in P_k. We see that $f_k^0 + f_k^1 = \sum X_j$, and $\Pr[X_j = 1] \geq r_k^n$, and since the probability that $f_k^0 + f_k^1$ is too small is maximized when $\Pr[X_j = 1] = r_k^n$, we assume that this is the case. Recall that $r_k^n \geq \frac{z(n,k)\rho}{e}$. By the assumption that $b \leq \lceil \log n \rceil$, this implies that $r_k^n \geq \frac{np}{2ed}$. The claim now follows from the second Chernoff bound above, using $\delta = \frac{1}{2}$. ■

We next demonstrate that our estimate of q_k^n is quite accurate:

Claim 3 *Given that* $f_k^0 + f_k^1 \geq \mu$, $\Pr\left[|\bar{q}_k^n - q_k^n| > \frac{\rho r^{z(n,k)}}{4e^2}\right] \leq \frac{\Delta}{4d}$.

Proof: Let $\bar{r}_k^n$ be the actual fraction of the F packets P which are reset by some node and $I_0^P = k$. Since $q_k^n = \frac{f_k^1 + f_k^0 - \bar{r}_k^n \cdot F}{f_k^1 + f_k^0}$, we see that $|\bar{q}_k^n - q_k^n| = \frac{\bar{r}_k^n \cdot F - r_k^n \cdot F}{f_k^1 + f_k^0}$. If we do not condition on $f_k^0 + f_k^1 \geq \mu$, then the fact that $\Pr\left[|\bar{r}_k^n \cdot F - r_k^n \cdot F| > \frac{\rho r^{z(n,k)}}{8e^2} r_k^n F\right] \leq \frac{\Delta}{5d}$ follows from the Chernoff bounds above, the fact that $r_k^n \geq \frac{\rho n}{2ed}$, and the assumption that $b \geq 2$. If we then condition on $f_k^1 + f_k^0 \geq \mu$, by Claim 2, this increases $\Pr\left[|\bar{r}_k^n \cdot F - r_k^n \cdot F| > \frac{\rho r^{z(n,k)}}{8e^2} r_k^n F\right]$ to at most $\frac{\Delta}{4d}$. Thus, with probability at most $\frac{\Delta}{4d}$, $|\bar{q}_k^n - q_k^n| > \frac{\rho r^{z(n,k)}}{8e^2} \frac{r_k^n F}{\mu} \geq \frac{\rho r^{z(n,k)}}{4e^2}$, where the second inequality again uses the fact that $r_k^n \geq \frac{\rho n}{2ed}$. ■

We next demonstrate what the implications of this are on our algorithm:

Claim 4

$$\left| p_k^0 - \sum_{j=1}^{z(n,k)} c_j^k B_{(j-1)d+k} \right| \leq |\bar{q}_k^n - q_k^n| - (|\bar{q}_k^n - q_k^n|) r^{z(n,k)}$$

Proof: Since $\alpha_j^k \leq 1$, we see that $\left| p_k^0 - \sum_{j=1}^{z(n,k)} c_j^k B_{(j-1)d+k} \right| \leq \sum_{j=1}^{z(n,k)} |\bar{q}_k^n - q_k^n| \frac{r^{j-1}}{2} \leq |\bar{q}_k^n - q_k^n| - (|\bar{q}_k^n - q_k^n|) r^{z(n,k)}$. ■

Claim 5 *Given that* $f_k^0 + f_k^1 \geq \mu$,

$$\Pr\left[|p_k - p_k^0| > \frac{\bar{q}_k^n}{2} r^{z(n,k)} + (|\bar{q}_k^n - q_k^n|) r^{z(n,k)} + \frac{\rho r^{z(n,k)}}{4e^2}\right] \leq \frac{\Delta}{2d}.$$

Proof: We here bound the probability that p_k is too large; the probability that p_k is too small is similar. It is easy to see that $\Pr\left[p_k - p_k^0 > \frac{\bar{q}_k^n}{2} r^{z(n,k)} + (|\bar{q}_k^n - q_k^n|) r^{z(n,k)} + \frac{\rho r^{z(n,k)}}{4e^2}\right]$ is maximized when the adversary sets all initial values of d_0 to 1, and thus we assume that the adversary does so. Note that this implies that $E[p_k] = p_k^1 - \frac{\bar{q}_k^n r^{z(n,k)}}{2} = p_k^0 + q_k^n r^{z(n,k)} - \frac{\bar{q}_k^n r^{z(n,k)}}{2} \leq p_k^0 + \frac{\bar{q}_k^n}{2} r^{z(n,k)} + (|\bar{q}_k^n - q_k^n|) r^{z(n,k)}$. We now let X_j, for $1 \leq j \leq t_k$, be a random variable, where $X_j = 1$ if the jth packet in P_k arrives to N_0 with $d_0 = 1$ and $X_j = 0$ otherwise, where $t_k = f_k^1 + f_k^0$. We shall bound the probability that $\sum_{j=1}^{t_k} X_j > t_k(p_k^1 + \frac{\rho r^{z(n,k)}}{4e^2})$.

Unfortunately, we can not use a Chernoff bound on this sum directly, since conditioning on $f_k^0 + f_k^1 \geq \mu$ can result in a small amount of dependence between the X_js (this is actually somewhat subtle). To remove this dependence, we partition the integers from 1 to t_k into two sets, where $j \in S_0$ if packet j arrives without being reset, and $j \in S_1$ otherwise. The variables X_j for $j \in S_0$ are independent, as are the variables X_j for $j \in S_1$. Let $s_0 = \Pr[X_j = 1]$ for $j \in S_0$. We see that $s_0 = r^{z(n,k)} + \sum_{j=1}^{z(n,k)} B_{k+(j-1)d} \frac{r^{j-1}}{2}$. Likewise, let $s_1 = \Pr[X_j = 1]$ for $j \in S_1$. We see that $s_1 = \sum_{j=1}^{z(n,k)} B_{k+(j-1)d} \alpha_j^k \frac{r^{j-1}}{2}$.

We show that for $w \in \{0, 1\}$, $\Pr\left[\sum_{j \in S_w} X_j > |S_w| s_w + t_k \frac{\rho r^{z(n,k)}}{8e^2}\right] \leq \Delta/8d$. Since $|S_0| = q_k^n t_k$ and $|S_1| = (1 - q_k^n) t_k$, this implies that $\Pr\left[\sum_{j=1}^{t_k} X_j > t_k(p_k^1 + \frac{\rho r^{z(n,k)}}{4e^2})\right] \leq \Delta/4d$, from which the claim follows. By the first Chernoff bound above,

$$\Pr\left[\sum_{j \in S_w} X_j > |S_w| s_w + t_k \frac{\rho r^{z(n,k)}}{8e^2}\right] \leq e^{-\left(\frac{t_k \rho r^{z(n,k)}}{8e^2 |S_w| s_w}\right)^2 \frac{|S_w| s_w}{3}}.$$

This probability is maximized by making $|S_w| s_w$ as large as possible, but it must be the case that $|S_w| s_w \leq t_k$. Thus, we may consider only the case where $|S_w| s_w = t_k$. Now, since we are conditioning $t_k \geq \mu$, and we have that $b \geq 2$, we see that

$$e^{-\left(\frac{t_k \rho r^{z(n,k)}}{8e^2 |S_w| s_w}\right)^2 \frac{|S_w| s_w}{3}} \leq \Delta/8d.$$

■

Now note that Claims 2, 3, 4, and 5 together give us that

$$\Pr\left[\left|p_k - \sum_{j=1}^{z(n,k)} c_j^k B_{(j-1)d+k}\right| > \frac{\bar{q}_k^n r^{z(n,k)}}{2} + \frac{\rho r^{z(n,k)}}{2e^2}\right] \leq \Delta/d.$$

Thus, we only have left to show that

$$\frac{\bar{q}_k^n r^{z(n,k)}}{2} + \frac{\rho r^{z(n,k)}}{2e^2} \leq \left(\bar{q}_k^n + (1 - \bar{q}_k^n)\alpha_{z(n,k)}^k\right) \frac{r^{z(n,k)}}{2},$$

or that $\frac{\rho}{e^2} \leq (1 - \bar{q}_k^n)\alpha_{z(n,k)}^k$. We have that

$$\alpha_{z(n,k)}^k \geq \frac{\rho(1-\rho)^{n-1}}{\sum_{t=1}^{z(n,k)} \rho},$$

and so using the assumption that $\rho \leq 1/n$, we see that $\alpha^k_{z(n,k)} \geq \frac{1}{ez(n,k)}$. Thus, we only need to show that $1 - \bar{q}^n_k \geq \frac{z(n,k)\rho}{e}$. By the definition of $\bar{q}^n_k$, this is equivalent to $f^1_k + f^0_k \leq \frac{e}{z(n,k)\rho} \cdot r^n_k \cdot F$. Since $r^n_k \geq \frac{z(n,k)\rho}{e}$, we only need that $f^1_k + f^0_k \leq F$. This follows simply from the fact that at most, all the packets are in the set P_k. ■

4 The lower bound

For our lower bound, we consider the simple adversary strategy where every packet the adversary sends simply has all bits set to 0. Thus, this lower bound really captures the difficulty of sending information from a memoryless node using a bounded number of bits. It is somewhat surprising that we are able to provide an algorithm that achieves close to this lower bound with the significantly more powerful adversary of the upper bound. Also recall that our lower bound model assumes a single intermediate node with all n bits of the input, rather than the more distributed model of the upper bound. With these assumptions, we are able to prove our lower bound.

For any protocol P, let $\mathcal{E}(\mathcal{P})$ be the expected number of packets received when the input is chosen uniformly at random from the set of all 2^n possible inputs. Let $p(\mathcal{P})$ be the probability that protocol $\mathcal{P}$ makes a mistake when the input is chosen uniformly at random from the set of all 2^n possible inputs.

Theorem 3 *For any protocol $\mathcal{P}$, if $\mathcal{E}(\mathcal{P}) \leq \frac{2^b-1}{8e}2^{n/2^b} - 2^{b-2}$, then $p(\mathcal{P}) \geq 1/2$.*

Proof: Any algorithm employed by the victim can be thought of as a (possibly randomized) procedure for deciding, for each possible sequence of packets that the victim has received, whether or not to continue receiving packets, and if the victim decides to not continue, then the procedure must specify a probability distribution over possible results for the victim to output. We refer to such an algorithm as a *general* protocol. A restricted class of protocols is *Monte Carlo* protocols, where the victim waits until it has received exactly T packets, where T depends only on n and b. The protocol maps the set of T received packets to a distribution over possible results, which the victim uses to produce an output.

Lemma 2 *For any general protocol $\mathcal{P}$, there is a Monte Carlo protocol $\mathcal{P}'$, such that $\mathcal{E}(\mathcal{P}') = 4\mathcal{E}(\mathcal{P})$, and $p(\mathcal{P}') \leq p(\mathcal{P}) + 1/4$.*

Proof: We define $\mathcal{P}'$ as follows: collect $T = 4\mathcal{E}(\mathcal{P})$ packets. Using the order that the packets arrive at the victim, simulate the protocol $\mathcal{P}$. If $\mathcal{P}$ produces a result before it receives T packets, then $\mathcal{P}'$ produces the same result, ignoring the remainder of the packet that it has. If $\mathcal{P}$ has not produced a result after receiving T packets, then $\mathcal{P}'$ specifies to the victim to output a result chosen uniformly at random from the set of all 2^n possible outputs. The bound on $p(\mathcal{P}')$ follows from the fact that by Markov's inequality, the probability that $\mathcal{P}$ has not produced a result after receiving T packets is at most $1/4$. ■

We shall demonstrate that for any Monte Carlo protocol, if the number of packets received is too small, then the probability that the protocol makes a mistake is at least $3/4$. This result, combined with Lemma 2, implies the theorem. Thus, we henceforth only consider Monte Carlo protocols.

The input to the victim can be described via a *receipt sequence*: a sequence $(r_1 \ldots r_T)$, where r_i is a b-bit string describing the ith b-bit packet that is received by the victim. Any Monte Carlo protocol for the victim is a function that maps a receipt sequence to a probability distribution over n-bit strings. Another kind of description of the input to the victim is a *receipt profile*: a 2^b-tuple $R = (r_0, \ldots, r_{2^b-1})$, where r_i is the number of packets of type i received by the victim. Note that $\sum_{j=0}^{2^b-1} r_j = T$. For any receipt profile R, let $S(R)$ be the set of receipt sequences S such that for all i, $0 \leq i \leq 2^b - 1$, the number of packets of type i in the sequence S is exactly r_i. Let a *permutation oblivious* algorithm for the victim be a function that maps a receipt profile to a probability distribution over n-bit strings. Intuitively, a permutation oblivious algorithm is a Monte Carlo algorithm that ignores the permutation information of the input, and only uses the receipt profile of the input.

Lemma 3 *For any Monte Carlo algorithm $\mathcal{P}'$ for the victim, there is a permutation oblivious algorithm $\mathcal{P}''$ for the victim, such that $\mathcal{E}(\mathcal{P}'') = \mathcal{E}(\mathcal{P}')$, and $p(\mathcal{P}'') = p(\mathcal{P}')$.*

Proof: Given a Monte Carlo algorithm $\mathcal{P}'$ for the victim, we define $\mathcal{P}''$ as follows: on an input receipt profile R, choose a receipt sequence S from $S(R)$ uniformly at random. The probability distribution over n-bit strings returned by $\mathcal{P}''$ is the same as $\mathcal{P}''$ would return when the input is S. To see that $p(\mathcal{P}'') = p(\mathcal{P}')$, note that since the intermediate node is memoryless, on any n-bit string that is input to the intermediate node, and for any receipt profile R, the probability that the receipt sequence is any receipt sequence in $S(R)$ is the same for all receipt sequences in $S(R)$. Thus, it does not matter whether the intermediate node "chooses" a receipt sequence uniformly at random from the set of receipt sequences in the receipt profile, or whether the victim makes this same choice. ■

Thus, we can simply show a lower bound for permutation oblivious algorithms, and this will imply a lower bound for all possible algorithms. Let $\psi(T)$ be the set of all possible receipt profiles for which the total number of packets received is exactly T. Let $\iota(n)$ be the set of all 2^n inputs to the intermediate node. For any $\tau \in \psi(T)$ and $I \in \iota(n)$, let $p(\tau, I)$ be the probability that the algorithm outputs I when the receipt profile is τ. Note that for any input I, the probability that the algorithm outputs I, given that the intermediate node receives the input I, is at most $\sum_{\tau \in \psi(T)} p(\tau, I)$. Thus, for any permutation oblivious algorithm $\mathcal{P}''$,

$$p(\mathcal{P}'') \geq \frac{\sum_{I \in \iota} \left(1 - \sum_{\tau \in \psi(T)} p(\tau, I)\right)}{2^n}.$$

Now, note that

$$\sum_{\tau \in \psi(T); I \in \iota(n)} p(\tau, I) \leq |\psi(T)|.$$

This implies that $p(\mathcal{P}'') \geq 1 - \frac{|\psi(T)|}{2^n}$. Thus, if $|\psi(T)| \leq 2^n/4$, the permutation oblivious protocol must make a mistake with probability at least $3/4$. Thus, we only need to compute $|\psi(T)|$ for a given value of b. By a standard combinatorial argument, the number of receipt profiles in $\psi(T)$ is simply

$$\binom{T + 2^b - 1}{2^b - 1} \leq \left(\frac{(T + 2^b - 1)e}{2^b - 1}\right)^{2^b - 1}.$$

Thus, $p(\mathcal{P}'') \geq 3/4$, provided that $\left(\frac{(T+2^b)e}{2^b-1}\right)^{2^b} \leq 2^n/4$, or that $T \leq \frac{2^b-1}{2e}2^{n/2^b} - 2^b$. By Lemmas 2 and 3, this implies that for any general protocol $\mathcal{P}$, $p(\mathcal{P}) \geq 1/2$, provided that $\mathcal{E}(\mathcal{P}) \leq \frac{2^b-1}{8e}2^{n/2^b} - 2^{b-2}$. ■

We also point out that a slightly tighter analysis of the required value of T gives us that when $b = 1$, the lower bound is $\Omega(2^n)$.

5 Applications of the Model

In this section, we demonstrate that the upper bound model actually leads to a quite general solution. In particular, it is easy to map our solutions to a number of scenarios. We demonstrate this by considering three scenarios here: the case where the underlying tree is not binary, the case where the routers are not given the information of which child forwarded them a given packet, and the case where the routers have no information about the network topology or routing strategy other than a unique ID. Our lower bound model is strong enough that for all of these other scenarios considered, our lower bound still applies.

First, consider the case where the victim knows the entire routing topology, and also is able to see which child it receives a given packet from, but the underlying tree is not binary. We use the following type of encoding of the path: for every node N of the tree, we represent what child of that node is the predecessor of N in the adversary's path of attack using a Shannon code, where the distribution used in the code is the distribution over the children of N when a node of the subtree rooted at N is chosen uniformly at random.

Claim 6 *The maximum number of bits required to represent any path using this scheme is at most $h + \log m$, where m is the number of nodes in the system, and h is the height of the tree.*

Proof: Let $C_1 \ldots C_\ell$ be the sequence of code words used along any path from the victim to a node. Let T_j be the number of children in the subtree rooted at the jth node of this path. Since we are using a Shannon code, $\frac{T_j}{T_{j+1}} \geq 2^{|C_j|-1}$. Since $T_h = 1$ for any path, we have that $T_1 = \prod_{j=1}^{h-1} \frac{T_j}{T_{j+1}}$, and so $T_1 \geq 2^{(\sum_{i=1}^{\ell} |C_j|)-\ell}$. Since $m \geq T_1$, $\log m \geq (\sum_{i=1}^{\ell} |C_j|) - \ell$, from which the claim follows. ■

To use a protocol for the upper bound model to actually compute this encoding, consider some router R on the path of attack with t children, and corresponding code words $C_1 \ldots C_t$. On a message received from child j, router R simulates what would occur in a tree of height $|C_j|$ when the child specified by C_j receives a packet from the adversary. This provides a mechanism for encoding the string C_j.

Next consider the case where nodes still know the entire routing topology, and the tree is binary, but nodes do not obtain the information of which child they received a given packet from. In this case, the information that is reconstructed is slightly different: instead of being able to reconstruct a path that contains the attacker, this smaller amount of information only allows us to reconstruct a path such that the attacker is a child of some node along that path. We refer to this model as the *source oblivious* model, and the model used for the bulk of the paper as the *source cognizant* model. We show that the two models are equivalent in the following sense:

Claim 7 *Any protocol for the source cognizant model for a tree of height n provides a protocol for the source oblivious model for a tree of height $n-1$. Any protocol for the source cognizant model for a tree of height n provides a protocol for the source oblivious model for a tree of height $n-1$.*

Proof: To simulate a protocol for the source cognizant model in the source oblivious model, each node simply does exactly what its parent would do in the source cognizant model. Each node clearly has enough information to do this. The parent of the attacker follows the source oblivious protocol correctly, and this simulates the parent of the parent of the attacker following the source cognizant protocol correctly. Thus, the path is followed correctly back to the parent of the attacker.

For the reverse simulation, each node N does exactly what its child N_c on the path would do in the source oblivious protocol on receiving the bits that N receives. In the case that N_c is actually the attacker, the node N simulates what N_c would do if it were not the attacker, but a child of N_c were the attacker. Since the node N has the information of which child it receives a packet from, N has enough information to perform such a simulation. Since the parent of the attacker simulates what the attacker would do in the source oblivious protocol if the attacker were on the path, but not an attacker, this gives the source cognizant protocol the ability to obtain a path that contains the attacker. ■

Finally, consider the case where neither the victim nor the routers have any information about the network topology, nor do they see the information of which child sends it a packet. Each router simply knows a unique router ID, and our task is to inform the victim of ℓ unique IDs of routers along the path of attack. These IDs might for example correspond to IP addresses. We assume that a router ID can be any k-bit string. To use a protocol for the upper bound model in such a scenario, each router, on receiving a b-bit string from a preceding router would simulate a binary tree of height k in the source oblivious model, where the node in the tree that receives the packet is the node corresponding to the string that represents the ID of the router.

References

[1] S. M. Bellovin, ICMP Traceback Messages. Internet Draft: draft-bellovin-itrace-00.txt, Mar. 2000.

[2] Hal Burch and Bill Cheswick, Tracing Anonymous Packets to Their Approximate Source. In *Proc. Usenix LISA '00*, 2000.

[3] Sven Dietrich, Neil Long, and David Dittrich, Analyzing Distributed Denial of Service Attack Tools: The Shaft Case. In *Proc. 14th Systems Administration Conference, LISA 2000.*

[4] Drew Dean, Matt Franklin, Adam Stubblefield, An Algebraic Approach to IP Traceback. In *Proc. 2001 Network and Distributed System Security Symposium.*

[5] P. Ferguson and D. Senie, RFC 2267: Network Ingress Filetring: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. *The Internet Society*, 1998.

[6] S. Floyd and V. Jacobson, Random Early Detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4), August 1997.

[7] S. Lee and C. Shields, Tracing the Source of Network Attack: A Technical, Legal and Societal Problem. In *Proceedings of the 2001 IEEE Workshop on Information Assurance and Security*, June 2001.

[8] K. Park and H. Lee. On the effectiveness of probabilistic packet marking for IP traceback under denial of service attack. In *Proc. IEEE INFOCOM '01*, pp. 338-347, 2001.

[9] K. Park and H. Lee. On the effectiveness of route-based packet filtering for distributed DoS attack prevention in power-law internets. To appear in *Proc. ACM SIGCOMM '01*, August 2001.

[10] C. Perkins, IP Mobility Support. RFC 2002, Oct. 1996.

[11] Rajeev Motwani and Prabhakar Raghavan, *Randomized Algorithms*. Cambridge University Press, New York, NY, 1995.

[12] Stefan Savage, David Wetherall, Anna Karlin and Tom Anderson, Practical Network Support for IP Traceback. In *Proceedings of ACM SIGCOMM 2000* , pp. 295-306, August 2000.

[13] Alex C. Snoeren, Craig Partridge, Luis A. Sanchez, Christine E. Jones, Fabrice Tchakountio, Stephen T. Kent, and W. Timothy Strayer. Hash-Based IP Traceback. To appear in *Proc. ACM SIGCOMM 2001*, August 2001.

[14] Dawn X. Song and Adrian Perrig, Advanced and authenticated marking schemes for IP traceback, In *Proc. IEEE INFOCOM '01*, 2001.