

# An Algebraic Approach to IP Traceback

Drew Dean	Matt Franklin	Adam Stubblefield
Xerox PARC	U.C. Davis	Rice University
ddean@parc.xerox.com	franklin@cs.ucdavis.edu	astubble@rice.edu

August 7, 2001

## Abstract

We present a new solution to the problem of determining the path a packet traversed over the Internet (called the *traceback problem*) during a denial of service attack. This paper reframes the traceback problem as a polynomial reconstruction problem and uses algebraic techniques from coding theory and learning theory to provide robust methods of transmission and reconstruction.

## 1 Introduction

A denial of service attack is designed to prevent legitimate access to a resource. In the context of the Internet, an attacker can “flood” a victim’s connection with random packets to prevent legitimate packets from getting through. These Internet denial of service attacks have become more prevalent recently due to their near untraceability and relative ease of execution [9]. Also, the availability of tools such as Stacheldraht [11] and TFN [12] greatly simplify the task of coordinating hundreds or even thousands of compromised hosts to attack a single target.

These attacks are so difficult to trace because the only hint a victim has as to the source of a given packet is the source address, which can be easily forged. Although ingress filtering can help by preventing a packet from leaving a border network without a source address from the border network [14], attackers have countered by choosing legitimate border network addresses at random. The traceback problem is also difficult because many attacks are launched from compromised systems, so finding the source of the attacker’s packets may not lead to the attacker. Disregarding the problem of finding the person responsible for the attack, if a victim was able to determine the path of the attacking packets in near real-time, it would be much easier to quickly stop the attack. Even finding out partial path information would be useful because attacks could be throttled at far routers.

This paper presents a new scheme for providing this traceback data by having routers embed information randomly into packets. This is similar to the technique used by Savage, *et al* [24], with the major difference being that our schemes are based on algebraic techniques. This has the advantage of providing a scheme that offers more

flexibility in design and more powerful techniques that can be used to filter out attacker generated noise and separate multiple paths. Our schemes share similar backwards compatibility and incremental deployment properties to the previous work.

More specifically, our scheme encodes path information as points on polynomials. We then use algebraic methods from coding theory to reconstruct these polynomials at the victim. This appears to be a powerful new approach to the IP traceback problem.

We note that although the study of traceback mechanisms was motivated by denial of service attacks, there are other applications as well. These methods might be useful for the analysis of legitimate traffic in a network. For example, congestion control, robust routing algorithms, or dynamic network reconfiguration might benefit from real-time traceback mechanisms.

The rest of the paper is organized as follows: Section 2 discusses related work, Section 3 contains an overview of the problem and our assumptions, Section 4 presents our approach for algebraically coding paths, Section 5 gives detailed specifications for some of our schemes, Section 6 provides a mathematical analysis of the victim's reconstruction task, Section 7 discusses the issue of encoding marking data in IP packets, and 8 gives conclusions and future work.

## 2 Related Work

The idea of randomly encoding traceback data in IP packets was first presented by Savage, *et al* [24]. They proposed a scheme in which adjacent routers would randomly insert adjacent edge information into the ID field of packets. Their key insight was that traceback data could be spread across multiple packets because a large number of packets was expected. They also include a distance field which allows a victim to determine the distance that a particular edge is from the host. This prevents spoofing of edges from closer than the nearest attacker. The biggest disadvantage of this scheme is the combinatorial explosion during the edge identification step and the few feasible parameterizations. The work of Song and Perrig provides a more in depth analysis of this scheme [25].

There have been many other notable proposals for IP traceback since the original proposal. Bellovin has proposed having routers create additional ICMP packets with traceback information at random and a public key infrastructure to verify the source of these packets [4]. This scheme can also be used in a non-authenticated mode, although the attackers can easily forge parts of routes that are farther from the victim than the closest of the attackers.

Song and Perrig have an improved packet marking scheme that copes with multiple attackers [25]. Unfortunately, this scheme requires that all victims have a current map of all upstream routers to all attackers (although Song and Perrig describe how such maps can be maintained). Additionally, it is not incrementally deployable as it requires all routers on the attack path to participate (although Song and Perrig note that it also suffices for the upstream map to indicate which routers are participating).

Doepfner, Klein, and Koyfman proposed adding traceback information to an IP option [13]. Besides the large space overhead, this solution would cause serious problems with current routers, as they are unable to process IP packets with options in hardware.

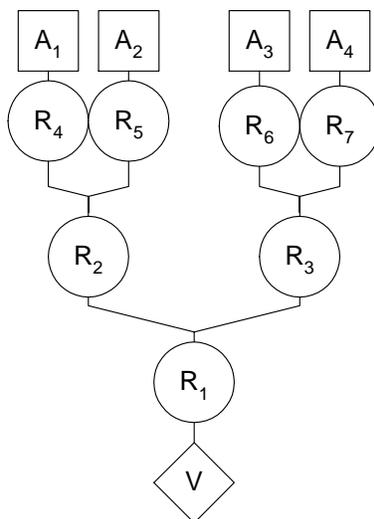


Figure 1: Our example network.

It also causes others issues, for example, adding the option may require the packet to be fragmented.

Burch and Cheswick have a scheme that uses UDP packets and does not require the participation of intermediate ISPs [8]. This scheme, however, assumes that the denial of service attack is coming from a single source network. This differs from us as we aim to distinguish multiple attacking hosts.

Lee and Park have analyzed packet marking schemes in general [19]. Their paper contains general tradeoffs between marking probability, recovered path length, and packets received, that can be applied to any of the probabilistic marking schemes, including the one in this paper.

We refer the reader to Savage’s paper for a discussion of other methods to detect and prevent IP spoofing and denial of service attacks.

The algebraic techniques we apply were originally developed for the fields of coding theory [15] and machine learning [2]. For an overview of algebraic coding theory, we refer the reader to the survey by Sudan [27] or the book by Berlekamp [6].

### 3 Overview

This paper addresses what Savage, *et al* call the *approximate traceback problem*. That is, we would like to recover all paths from attacker to victim, but we will allow for paths to have invalid prefixes. For example, for the network shown in Figure 1, the true path from the attacker  $A_1$  to the victim  $V$  is  $R_4R_2R_1$ . We will allow our technique to also produce paths of the form  $R_2R_6R_4R_2R_1$  because the true path is a suffix of the recovered path.

Our family of algebraic schemes was motivated by the same assumptions as used in previous work:

1. Attackers are able to send any packet
2. Multiple attackers can act together
3. Attackers are aware of the traceback scheme
4. Attackers must send at least thousands of packets
5. Routes between hosts are in general stable, but packets can be reordered or lost
6. Routers can not do much per-packet computation
7. Routers are not compromised, but not all routers have to participate

## 4 Algebraic Coding of Paths

We will now present a series of schemes that use an algebraic approach for encoding traceback information. All of these schemes are based on the principal of reconstructing a polynomial in a prime field. The basic idea is that for any polynomial  $f(x)$  of degree  $d$  in the prime field  $GF(p)$ , we can recover  $f(x)$  given  $f(x)$  evaluated at  $(d+1)$  unique points. Let  $A_1, A_2, \dots, A_n$  be the 32-bit IP addresses of the routers on path  $P$ . Let  $f_P(x) = A_1x^{n-1} + A_2x^{n-2} + \dots + A_{n-1}x + A_n$ . We associate a packet id  $x_j$  with the  $j$ th packet. We then somehow evaluate  $f_P(x_j)$  as the packet travels along the path, accumulating the result of the computation in a running total along the way. When enough packets from the same path reach the destination, then  $f_P$  can be reconstructed by interpolation. The interpolation calculation might be a simple set of linear equations, if all of the packets received at the destination traveled the same path. Otherwise, we will need to employ more sophisticated interpolation strategies that succeed even in the presence of incorrect data or data from multiple paths [5, 28, 15, 7]. These methods were developed originally for use in coding theory and learning theory.

A naive way to evaluate  $f_P(w)$  would be to have the  $j$ th router add  $A_jw^{n-j}$  into an accumulator that kept the running total. Unfortunately, this would require that each router know its position in the path and the total length of the path. We could eliminate the need for each router to know the total length of the path (while still requiring each router to know its position in the path) by reordering the coefficients of  $f_P$ :  $A_1 + A_2w + A_3w^2 + \dots + A_nw^{n-1}$ . However, we can do even better by sticking with our original ordering, and using an alternative means of computing the polynomial. Specifically, to compute  $f_P(w)$ , each router  $R_j$  multiplies the amount in the accumulator by  $w$ , adds  $R_j$ , and returns the result to the accumulator, and passes the packet on to the next router in the path (Horner's rule [18]). For example,  $((((0 \cdot w) + R_1)w + R_2)w + R_3)w + R_4 = R_1w^3 + R_2w^2 + R_3w + R_4$ . Notice that the router doesn't need to know the total length of the path or its position in the path for this computation of  $f_P$ .

## 4.1 Deterministic Path Encoding

The simplest scheme that uses this algebraic technique encodes an entire path. At the beginning of a path, let  $FullPath_{0,j} = 0$ . Each router  $i$  on the path calculates  $FullPath_{i,j} = (FullPath_{i-1,j} \cdot x_j + R_i) \bmod p$  where  $x_j$  is a random value passed in each packet,  $R_i$  is the router's IP address and  $p$  is the smallest prime larger than  $2^{32} - 1$ . The value  $FullPath_{i,j}$  is then passed in the packet, along with  $x_j$ , to the next router. At the packet's destination  $FullPath$  will equal  $(R_n x^{n-1} + R_{n-1} x^{n-2} + \dots + R_2 x + R_1) \bmod p$ , which can be reconstructed by solving the following matrix equation over  $GF(p)$ :

$$\begin{pmatrix} 1 & x_1 & x_1^2 & \dots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \dots & x_2^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^{n-1} \end{pmatrix} \begin{pmatrix} R_1 \\ R_2 \\ \vdots \\ R_n \end{pmatrix} = \begin{pmatrix} FullPath_{n,1} \\ FullPath_{n,2} \\ \vdots \\ FullPath_{n,3} \end{pmatrix}$$

As long as all of the  $x_i$ 's are distinct, the matrix is a Vandermonde matrix (and thus has full rank) and is solvable in  $O(n^2)$  field operations [22].

Assuming that we get a unique  $x_j$  in each packet, we can recover a path of length  $d$  with only  $d$  packets. The downside, however, is that this scheme would require  $\log_2(p) + 2\lceil \log_2(d) \rceil$  bits per packet (the first term is the encoding of the running  $FullPath$  and the second term is the encoding of the  $x_j$  and  $y_j$  values). Even for modest maximum path lengths of 16, the space required (68 bits, counting 4 bits for recording the number of routers in the path, and 32 bits each for the  $x$  coordinate and  $y$  coordinate of the point on the polynomial) far exceeds the number of bits available to us in an IP header.

We could split a router's IP address into  $c$  chunks and add  $\lceil \log_2(c) \rceil$  bits to indicate which chunk was represented in a given packet. Another approach would be to have each router add all of its chunks into each packet. That is, each router would update  $FullPath$   $c$  times, substituting each chunk of their IP address in order. The destination could then trivially reconstruct the IP addresses by interpolating to recover  $\tilde{f}_P(x) = R_{1,1} + R_{1,2}x + \dots + R_{1,c}x^{c-1} + R_{2,1}x^c + \dots + R_{n,c}x^{nc-1}$ , where  $R_{j,1}, R_{j,2}, \dots, R_{j,c}$  are the successive chunks of  $R_j$ . This would increase the degree of  $f$  by a factor of  $c$ , which would impact the performance of the reconstruction algorithm.

## 4.2 Randomized Path Encoding

In the above schemes, we require  $FullPath_{0,j} = 0$ . This implies that there is some way for a router to know that it is the "first" participating router on a particular path. In the current Internet architecture there is no reliable way for a router to have this information. We must therefore extend our scheme to mitigate this problem.

In our revised scheme a router first flips a weighted coin. If it came up tails the router would assume it was not the first router and simply follow the  $FullPath$  algorithm presented above, adding its IP address (or IP address chunk) data. On the other hand, if the coin came up heads, the router would assume it was the first router and randomly choose a  $x_j$  to use for the path. We will refer to this state as "marking mode."

This overall approach – which might be called the “reset paradigm” – was also used by Savage et al. for their traceback solutions.

At the destination, we would receive a number of different polynomials, all representing suffixes of the full path. In our example network, packets from  $A_1$  could contain  $R_4R_2R_1$ ,  $R_2R_1$ , or  $R_1$ .

We could change our marking strategy slightly. Whenever a router receives a packet, it still flips a weighted coin. But now, instead of simply going into marking mode for one packet when the coin comes up heads, the router could stay in marking mode for the next  $\tau$  packets it receives. More generally, the reset behavior could follow any Markov Process.

One problem is that attackers can cause more false paths than true paths to be received at the victim. This is due to the fact the our choice of a small  $p$  creates large number of packets in which no router on the packet’s path is in marking mode. The attacker can thus insert any path information he wishes into such packets. Because the attacker can generally find out the path to his victim (using *traceroute*, for example) he can compute  $FullPath_{0,j} = (FakePath_j/x_j^n - R_n x_j^{n-1} - \dots - R_0) \bmod p$ . This choice will cause the victim to receive  $FullPath_j = FakePath_j$ . When trying to reconstruct paths, the victim will have no indication as to which paths are real and which paths are faked. Two solutions to this problem are to increase  $p$  or to store a hop count (distance field) in the packet that each participating router would increment. Increasing the probability makes it even harder to receive long paths. Adding a hop count would prevent an attacker from forging paths (or suffixes of paths) that are closer than its actual distance from the victim but would require  $\lceil \log_2(d) \rceil$  more bits in the packet.

Our schemes could also make use of the HMAC techniques discussed by Song and Perrig to ensure that edges are not faked, but this would require us to either use additional space in the packets to store the hash or lose our incremental deployment properties [25]. If we decided to make one of these tradeoffs, our scheme would be comparably secure against multiple attackers.

### 4.3 Edge Encoding

We could add another parameter,  $\ell$ , that represents the maximum length of an encoded path. The value of  $\ell$  is set by the marking router and decremented by each participating router who adds in their IP information. When the value reaches 0, no more routers add in their information. For example, in the full path encoding scheme  $\ell = \infty$ , while  $\ell = 1$  would represent encoding of edges between routers. When  $\ell = 1$ , we call this an “algebraic edge encoding” scheme.

The benefit of this change would be to decrease the maximum degree  $d$  of the polynomials in order to reduce the number of packets needed out of a given set or packets to recover a route. The cost of this change is that it would add  $\lceil \log_2(\ell + 1) \rceil$  bits to the packets.

Of course, if  $\ell$  is less than the true path length, then reconstruction finds arbitrary subsequences of the path (not just suffixes as in Full Path encoding). The victim still has some work to do to combine these subsequences properly (as described in Savage et al.). Thus reconstruction in this scheme has an algebraic step followed by a combinatorial step.

## 5 Pseudocode for Sample Algebraic Schemes

In this section, we present pseudocode for some sample algebraic marking schemes that are based on the principles described in the previous section. Recall that each router has a unique 32-bit id.

### 5.1 Algebraic Edge Encoding

Here is the router's pseudocode for Edge1, an algebraic edge encoding scheme. Each packet is marked with  $32 + \lceil \log n \rceil + 1$  bits, where  $n$  is the number of  $x$  values. The degree of the polynomial is one.

```
Edge1 Marking procedure at router R:
  for each packet w
    with probability p
      w.xval := random;
      w.yval := R;
      w.flag := 1;
    otherwise if w.flag = 1 then
      w.yval := w.yval * w.xval + R
      w.flag := 0
```

Here is Edge2, algebraic edge encoding with  $c$  “chunks” per “hop”. Each packet is marked with  $\lceil 32/c \rceil + \lceil \log n \rceil + 1$  bits. The degree of the polynomial is  $2c - 1$ .

```
Edge3 Marking procedure at router R:
  for each packet w
    with probability p
      w.xval := random;
      w.yval := R[c] w.xval^{c-1} + R[c - 1] w.xval^{c-2}
              + .. + R[1];
      w.flag := 1;
    otherwise if w.flag = 1 then
      w.yval := w.yval * w.xval^c + R[c] w.xval^{c-1} +
              R[c - 1] w.xval^{c-2} + ... + R[1];
      w.flag := 0
```

Here is Edge3, which is identical to Edge2 except that each packet also has a distance field (“hop count”). Following Savage et al., we reserve five bits for the distance field. Each packet is marked with  $\lceil 32/c \rceil + \lceil \log n \rceil + 6$  bits. The degree of the polynomial is  $2c - 1$ .

```

Edge3 Marking procedure at router R:
  for each packet w
    with probability p
      w.xval := random;
      w.yval := R[c] w.xval^{c-1} + R[c - 1] w.xval^{c-2}
                + .. + R[1];
      w.flag := 1;
      w.dist := 0;
    otherwise if w.flag = 1 then
      w.yval := w.yval * w.xval^c + R[c] w.xval^{c-1} +
                R[c - 1] w.xval^{c-2} + ... + R[1];
      w.flag := 0;
      w.dist := w.dist + 1

```

Here is Edge4, which is identical to Edge3 except that the second router only contributes half of the bits of its router id. This lowers the degree of the polynomial, and introduces a little uncertainty into the reconstruction process (if two routers at the same distance from the victim had router id's that agreed on all of the contributed bits). Each packet is marked with  $\lceil 32/c \rceil + \lceil \log n \rceil + 6$  bits. The degree of the polynomial is  $1.5c - 1$ .

```

Edge3 Marking procedure at router R:
  for each packet w
    with probability p
      w.xval := random;
      w.yval := R[c] w.xval^{c-1} + R[c - 1] w.xval^{c-2}
                + .. + R[1];
      w.flag := 1;
      w.dist := 0;
    otherwise if w.flag = 1 then
      w.yval := w.yval * w.xval^{c/2} + R[c/2] w.xval^{c/2-1} +
                R[c/2 - 1] w.xval^{c/2-2} + ... + R[1];
      w.flag := 0;
      w.dist := w.dist + 1

```

## 5.2 Algebraic Full Path Encoding

Here is the router's pseudocode for Full1, the full path encoding scheme. Each packet is marked with  $32 + \lceil \log n \rceil$  bits, where  $n$  is the number of possible  $x$  values. The degree of the path polynomial is at most  $L$ , the length of the path.

Full1 Marking procedure at router R:

```

for each packet w
  with probability p
    w.xval := random;
    w.yval := 0;
    w.yval := w.yval * w.xval + R

```

Here is the router’s pseudocode for Full2, the full path encoding scheme with a distance field (“hop count”). Following Savage, we reserve five bits for the distance field, so each packet is marked with  $37 + \lceil \log n \rceil$  bits.

```

Full2 Marking procedure at router R:
for each packet w
  with probability p
    w.xval := random;
    w.yval := 0;
    w.dist := 0;
    w.yval := w.yval * w.xval + R;
    w.dist := w.dist + 1

```

## 6 Path Reconstruction by the Victim

In this section, we look more closely at the problem of path reconstruction by the victim. Let  $k$  denote the number of attack paths. Let  $L$  denote the expected length of an attack path. For simplicity, we will assume that all attack paths are very close to  $L$  in length.

For the main scheme of Savage et al. (which uses a total of 16 bits), the complexity of path reconstruction by the victim is  $O(Lk^8)$ . The exponent of eight reflects a combinatorial task that the victim must try by brute force. Of course, if they had more room to work with in their marking scheme, then the reconstruction complexity would go down. For example, if they used 23 bits for their marking scheme (and divided the “padded” router id into four 16-bit chunks), then the victim’s reconstruction task reduces to  $O(Lk^4)$ .

Our goal is to design algebraic schemes that improve on the reconstruction complexity of Savage et al. There are two main algebraic reconstruction approaches that we consider:

**Reed-Solomon List Decoding:** Given  $(x_1, y_1), \dots, (x_N, y_N)$  distinct points, find all polynomials of degree at most  $d$  that pass through at least  $m$  of these points. Guruswami-Sudan give an algorithm to solve this problem in time  $O(N^3)$  when  $N < m^2/d$ . An improvement by Olshevsky and Shokrollahi reduces the time to  $O(N^{2.5})$ .

More precisely, the reconstruction algorithm due to Guruswami and Sudan [15] can be implemented in a number of ways. The most straightforward implementation would take time  $O(n^{3d})$  to recover all edges for which we received at least  $\sqrt{dn}$  out of

$n$  packets. However, this drops to  $O(n^3)$  time by requiring only slightly more packets:  $\sqrt{dn(1+\delta)}$  out of  $n$ , for any  $\delta \geq 1$ . By scaling  $\delta$  appropriately, this allows us to trade off computation time (and memory) for accuracy. A recent algorithmic breakthrough by Olshevsky and Shokrollahi would reduce our reconstruction time even further, to  $O(n^{2.5})$  [21]. Moreover, this new algorithm is highly parallelizable (to up to  $O(n)$  processors), which suggests that distributing the reconstruction task might speed things up even more.

**Noisy Polynomial Interpolation:** Given  $(x_1, S_1), \dots, (x_n, S_n)$ , where each  $S_i$  has size at most  $m$ , find all polynomials  $f$  of degree at most  $d$  such that  $f(x_i) \in S_i$  for all  $i$ . Bleichenbacher-Nguyen give an algorithm to solve this problem whenever  $m < n/d$ , with running time identical to the Reed-Solomon List Decoding problem. They give other algorithms that work even when the bound  $m < n/d$  is not met.

**Types of Packets:** Let us assume that each packet that the victim receives is one of three possible types. A “true packet” contains a point on a polynomial that corresponds to a real attack path. A “bogus packet” contains a point created by an attacker outside the periphery, and never reset by any honest router along an attack path. A “stray packet” contains a point on a polynomial that corresponds to normal non-attack traffic. When a denial of service attack is underway, we assume that the fraction of stray packets is very small compared to true and bogus packets.

**False Positives:** A “false positive” is a polynomial that is recovered by the reconstruction algorithm, but does not correspond to part of an actual attack path. For Reed-Solomon list decoding, the expected number of false positives in a random sample is about  $(N!/(m!(N-m)!)) * (1/q)^{m-d-1}$ . For noisy polynomial interpolation, the expected number of false positives in a random sample is about  $m^n/q^{n-d-1}$ . For the main scheme of Savage et al., the expected number of false positives is about  $m^8/2^{32}$ .

When the marking scheme has no distance field, then we must also be concerned with “bogus edges” or “bogus paths” that the attacker can cause to appear in our sample. We will consider this separately from the issue of false positives that arise at random.

A moderate number of false positives is not a serious problem. Consider our marking scheme Edge3. The victim reconstructs a set of candidate edges for each distance. Each set of candidate edges includes true edges and “false positive” (but no “bogus edges” from the attacker assuming that no attacker is within this distance of the victim). Now the victim attempts to assemble paths by connecting edges from distance  $\ell$  with edges from distance  $\ell + 1$ . There is certainly no problem unless the first endpoint of a false positive edge from some distance  $\ell$  matches the second endpoint of a false positive or true edge from distance  $\ell + 1$ .

Let  $f$  be the expected number of false positives at each distance, and let  $k$  be the number of true edges at each distance. Then there are  $f$  expected false positives at distance  $\ell$ , and  $f + k$  expected false positives and true edges at distance  $\ell + 1$ . Let  $M$  be the number of distinct router id’s or partial router id’s that are possible (e.g.,  $2^{32}$  for Edge3). The probability of an accidental match is less than  $1 - ((M - f)/M)^{f+k}$  which is very close to  $1 - e^{-f(f+k)/M}$ . (When  $f$  is close to  $\sqrt{M}$ , this probability is unacceptably high.) The probability of an accidental match at any distance is less than  $1 - Le^{-f(f+k)/M}$  where  $L$  is the length of the longest path.

We now analyze the effectiveness of these approaches to path reconstruction. For each approach, the best known algorithms impose constraints on the design parameters for our marking schemes. It will be convenient for us to consider separately marking schemes that have a distance field and marking schemes that do not.

## 6.1 Reconstruction With a Distance Field

When the marking scheme has a distance field, the task of the victim is simplified. The victim can select a sample of packets for which  $w.\text{dist} = \ell$ , for any given  $\ell$ . As long as no attacker is within distance  $\ell$  of the victim, this sample will contain only true packets with points on polynomials that were last reset by routers at distance  $\ell$ .

### 6.1.1 Guruswami-Sudan Reconstruction With a Distance Field

The path reconstruction problem faced by the victim can be viewed as a Reed-Solomon list decoding problem. The distinct points are chosen from a random sample of the distinct points in packets that reach the victim.

The victim can filter out packets that were last reset at distance  $\ell$ , for every  $\ell$ . This simplifies the Reed-Solomon list decoding problem, by creating a smaller problem instance for each distance. We need  $N = nk$  packets from distance  $\ell$  to have  $n$  distinct points from each of  $k$  polynomials. The victim collects the largest possible sample of distinct points from packets with  $w.\text{dist} = \ell$  for every  $\ell$ . We need  $N < n^2/d$  to reconstruct the polynomials using the Guruswami-Sudan algorithm. Lastly, we need  $N^{2.5} < k^8$  for the efficiency of reconstruction to improve on Savage et al.

This has at least a few solutions, but the improvements are not so compelling. For example, using Edge3 with three 11-bit chunks can be competitive with Savage et al. for certain values of  $k$ .

### 6.1.2 Bleichenbacher-Nguyen Reconstruction With a Distance Field

The problem faced by the victim can be viewed as a noisy polynomial interpolation problem. The values  $x_1, \dots, x_n$  are all of the possible  $x$  values. Each set  $S_i$  contains all of the distinct  $y$  values such that  $(x_i, y)$  occurs in some packet within a random sample of all received packets. The polynomial  $f$  could be any of the polynomials that corresponds to a true attack path or a stray path.

The victim could proceed as follows. He looks at a sample of  $N$  packets (for suitably large  $N$ ), and for each  $x_i$  he chooses a set  $S_i$  of size  $m$  from all of the  $(x_i, y)$  points in the sample. If the number of distinct  $y$  values for which  $(x_i, y)$  occurs in the sample is greater than  $m$ , then the victim chooses which  $m$  values to include in  $S_i$  at random.

The victim can filter out packets that were last reset at distance  $\ell$ , for every  $\ell$ . This creates a smaller problem instance for each distance. For each problem instance, the number of  $S_i$  sets is equal to  $n$ , the number of possible  $x$  values. The size of each  $S_i$  is  $k$ , the number of attack paths. The degree of each polynomial is at most  $d$ , which depends on the particular algebraic encoding method we are using.

**False Positives:** There are  $k^n$  ways of taking one  $x$  value from each set. Each of these will actually be a polynomial of degree  $d$  or less with probability at most  $1/q^{n-d-1}$ .

Here  $q$  is the size of the finite field, which is essentially the number of distinct  $y$  values. We need the expected number of false positives  $k^n/q^{n-d-1}$  to be reasonably small.

For the basic reconstruction algorithm of Bleichenbacher-Nguyen, we need  $k < n/d$ . Three other algorithms by Bleichenbacher-Nguyen work for many  $k, n, d$  even if they do not satisfy  $k < n/d$ .

- A “meet-in-the-middle” algorithm has running time  $(n-d)m^{n/2}$  with precomputation that uses memory which is  $O(m^{n/4} \log q)$ . Note that this is independent of  $k$ .
- A “Grobner basis reduction” algorithm computes a Grobner basis reduction on a system of  $k$  polynomial equations in  $d+1$  unknowns. The best known Grobner basis algorithms are super-exponential in  $d$ , but reasonably efficient for small  $d$  (e.g.,  $d < 20$ ).
- A “lattice basis reduction” algorithm performs a lattice basis reduction on an  $(nk - n + 1)$ -dimensional lattice in  $Z^{nk}$  over a finite field of size about  $n$ . This method will be ineffective for our application because the size of the finite field is too small.

For efficiency over Savage et al., we need  $(nk)^{2.5} < k^8$ . Here are some interesting instantiations of our schemes with respect to this method of reconstruction:

**Example 1:** Edge3 encoding with 12 distinct  $x$  values (represented in a 4-bit  $xval$  field) and 8-bit  $yval$  field. Then the noisy polynomial problem has 12  $S_i$  sets, where the size of each  $S_i$  is  $k$ , the number of attack paths. The degree of each polynomial is at most 7. The size of the finite field is 256. The meet-in-the-middle algorithm takes time  $8k^6$ , which compares favorably to the  $k^8$  required by Savage et al. The Grobner basis reduction algorithm should also be reasonably efficient here. The total size of this marking scheme is 18 bits. However, the number of false positives is unacceptably high here:  $k^{12}/2^{32}$ .

**Example 2:** Edge4 encoding with 12 distinct  $x$  values (in a 4-bit  $xval$  field) and 8-bit  $yval$  field. Then the degree of each polynomial is at most 5. The running time for the meet-in-the-middle algorithm is about  $8k^6$ . The running time for the Grobner basis reduction algorithm is faster than in the previous example. The expected number of false positives is lower than in the previous example:  $k^{12}/2^{48}$ . If  $k = 16$ , then we expect about one false positive at each distance. (Of course, the risk from false positives is slightly greater than in the previous case, because the number of possible partial router id’s is only  $2^{22}$ . Thus there will be slightly more accidental matches of endpoints involving a bogus edge, but it is not significantly worse.) The total size of this marking scheme is 18 bits.

**Example 3:** Edge4 encoding with  $n = 10$  (4-bit  $xval$ ) and 11-bit  $yval$  field ( $q = 2^{11}$ ). Then the degree of each polynomial is at most 4 (using 22 bits of second router id). Running time for meet-in-the-middle is about  $6k^5$ , versus  $k^8$  for Savage et al. Number of false positives is about  $k^{10}/2^{55}$  versus  $k^8/2^{32}$  for Savage. For example, this is expected  $2^{-5}$  false positives for  $k = 32$ , or 32 false positives for  $k = 64$ , which are quite manageable. The total size of this marking scheme is 21 bits.

**Example 4:** Edge4 encoding with  $n = 8$  (3-bit xval) and 11-bit yval ( $q = 2^{11}$ ). Degree of each polynomial at most 4. Running time for meet-in-the-middle is about  $4k^4$ . Number of false positives is about  $k^8/2^{33}$  (e.g.,  $1/2$  when  $k = 16$ , or  $128$  when  $k = 32$ ). The total size of this marking scheme is 20 bits.

**Example 5:** Edge4 encoding with  $n = 12$  (4-bit xval) and 11-bit yval ( $q = 2^{11}$ ). Degree of each polynomial at most 4. Running time for meet-in-the-middle is about  $8k^6$ . Number of false positives is about  $k^{12}/2^{77}$  (e.g.,  $2^{-5}$  when  $k = 64$ , or  $128$  when  $k = 128$ , both of which are quite manageable). The total size of this marking scheme is 21 bits.

## 6.2 Reconstruction Without a Distance Field

For reconstruction when the marking scheme does not have a distance field, we do not achieve schemes that are competitive with Savage et al. Our analysis will begin with some facts and simplifying assumptions about the distribution of received packets by the victim.

### 6.2.1 Distribution of Received Packets

Let  $B_i$  be the fraction of packets arriving on the  $i$ th attack path that reach the victim as bogus packets. Let  $T_i$  be the fraction of packets on the  $i$ th attack path that reach the victim as true packets. Let  $F_i$  be the fraction of packets on the  $i$ th attack path that reach the victim as true packets that were only reset by the furthest router on that path. By Assumption 1,  $B_i + T_i = 1$ .

For all of the encoding schemes (unless “marking mode” is used), we have  $F_i = p(1-p)^{L-1}$ . Viewed as a function of  $p$  over  $[0 \dots 1]$ , this fraction takes on its maximum value at  $p = 1/L$ . When  $p = 1/L$ , this implies that  $F_i = 1/(e(L-1))$ .

For all of the encoding schemes, we have  $B_i = (1-p)^L$ . Then  $B_i = F_i(1-p)/p$ . When  $p = 1/L$ , this implies that  $B_i = 1/e$  and  $T_i = (L-1)F_i$ . The fact that there can be such a large fraction of bogus packets arriving on each path has serious consequences for our marking schemes without a distance field.

Let  $B, T, F$  be the fractions of bogus packets, true packets, and furthest packets for all paths to the victim. If we assume that the arrival rate of packets on all attack paths is approximately the same, then  $B_i = B$ ,  $T_i = T$ , and  $F_i = F$  for all  $i$ .

When “marking mode” is used, the probability that a router is not in reset mode is  $q = (1-p)^\tau$ . Then  $F_i = (1-q)q^{L-1}$ , and  $B_i = (1-q)^L$ .

**Coupon Collector’s Bound:** A sample of  $\lambda C \log C$  elements, drawn with replacement from  $[1 \dots C]$  according to the uniform distribution, is very likely to contain all  $C$  possible values, for some small constant  $\lambda$ .

### 6.2.2 Guruswami-Sudan Reconstruction Without a Distance Field

The victim can choose a random sample of distinct points in the packets that reach him. Without a distance field, he cannot partition the packets into smaller samples by last reset distance.

Assume that the routers are using an edge encoding scheme, and assume that we succeed if we can reconstruct all of the furthest edge polynomials. Let us also assume that we will search for polynomials that pass through  $n$  distinct points, where  $n$  is the number of distinct  $x$  values.

There are actually three distinct levels of reconstruction success that can be considered: (a) The sample of  $N$  points contains at least  $n$  points on every furthest edge polynomial with overwhelming probability; (b) The sample of  $N$  points contains at least  $n$  points on some furthest edge polynomial with overwhelming probability; (c) The sample of  $N$  points contains at least  $n$  points on some furthest edge polynomial with non-negligible probability  $q$ .

For case (a), the Guruswami-Sudan algorithm needs to be applied only once to a random sample of  $N$  points. For case (b), the algorithm needs to be applied  $\lambda k \log k$  times to independent random samples of  $N$  points (coupon collector's problem on the set of  $k$  furthest edge polynomials). For case (c), the algorithm needs to be applied  $\lambda k \log k / q$  times to independent random samples of  $N$  points.

For case (a), it suffices to have  $N \geq (\lambda n k \log(nk)) / (p(1-p)^{L-1})$ . That is because we are very likely to get a complete set of all  $n$  possible  $x$  values for all  $k$  edge polynomials. This implies that  $\lambda n k \log(nk)$  "samples" are sufficient. By the analysis of the preceding subsection, a "sample" from a furthest edge polynomial is expected in  $1 / (p(1-p)^{L-1})$  fraction of all of the packets. When combined with the Guruswami-Sudan bound, we get  $n^2 / d > (\lambda n k \log(nk)) / (p(1-p)^{L-1})$ . Assuming that  $p = 1/L$ , we have

$$n / \lambda \log(nk) > \lambda dk(L-1)e.$$

For case (b), it suffices to have  $N \geq M_{n,k} / (p(1-p)^{L-1})$ , where  $M_{n,k}$  is the answer to the following "occupancy problem": Throw  $M_{n,k}$  balls into  $k$  bins, and expect to find  $\lambda n \log n$  balls in the bin with the most balls. By the Pigeonhole Principle, it is certainly true that  $M_{n,k} < \lambda n k \log n$ . In fact, the actual value for  $M_{n,k}$  is quite close to this. Combined with the Guruswami-Sudan bound, we get  $n^2 / d > \lambda n k \log n / (p(1-p)^{L-1})$ . Assuming that  $p = 1/L$ , we have

$$\lambda n / \log n > dk(L-1)e.$$

For case (c), we can reduce the value of  $M_{n,k}$  a little, but it doesn't appear to be significant for our purposes.

Of course, for any of (a), (b), or (c), we can reduce  $N$  by eliminating from the sample any duplicate points. Since as many as  $1/e$  of all packets in the sample could be bogus packets from the attacker, removing duplicate points will have limited benefit. We can find no solution that yields a marking scheme that is more efficient than Savage et al. Moreover, for any plausible instantiation, the number of false positives and bogus edges (or bogus paths) is unacceptably high.

### 6.2.3 Bleichenbacher-Nguyen Reconstruction Without a Distance Field

The victim can proceed as described at the start of Section 6.1.2, although without a distance field the packets cannot be partitioned by last reset distance.

Version	H. Length	Type of Service (8-bit)	Total Length	
Fragment ID (16-bit)			(1-bit) Flags	Fragment Offset
Time to Live	Protocol	Header Checksum		
Source IP Address				
Destination IP Address				

Figure 2: The IP Header. Darkened areas represent underutilized bits.

Suppose that  $N$  is chosen to be large enough that all of the points on some furthest polynomial are included with high probability. The probability that a given  $S_i$  includes a point from  $f$  is at least  $m/n$ . The probability that reconstruction succeeds is at least  $(m/n)^n$ . The basic algorithm of Bleichenbacher and Nguyen solves the noisy polynomial interpolation whenever  $m < n/k$ .

This approach does not seem too promising when  $k > 1$ . In this case,  $(m/n)^n < 2^{-n}$ . Thus reconstruction is unlikely to succeed.

When  $k = 1$ , the victim can choose  $m = n - c$  for some positive integer  $c$ . The probability that reconstruction succeeds is at least  $(1 - c/n)^n$  which is about  $e^{-c}$ . Unfortunately, either the number of false positives is unacceptably large, or the success probability is unacceptably small.

Another approach would be to have the victim bias his sample with respect to how frequently different points occurred in the packets that reached him. Unfortunately, this does not appear to work well either. Since  $B_i = (L - 1)F_i$ , the victim will not be able to recognize the true packets that contain points from the furthest polynomials.

We conclude that when the marking scheme does not have a distance field, we do not see how to use the Bleichenbacher-Nguyen method of polynomial reconstruction, at least using their simplest algorithm. It is possible that their other algorithms, e.g., based on Grobner basis reduction, might be more effective.

## 7 Encoding Path Data

We now need a way to store our traceback data in IP packets. We will try to maximize the number of bits available to us while preserving (for the most part) backwards compatibility.

### 7.1 IP options

An IP option seems like the most reasonable alternative for storing our path information. Unfortunately, most current routers are unable to handle packets with options in

hardware [3]. Even if future routers had this ability, there are a number of problems associated with this approach as presented by Savage, *et al* [24]. For all of these reasons we have concluded that storing data in an IP option is not feasible.

## 7.2 Additional Packets

Instead of trying to add our path data to the existing IP packets, we could instead send the data out of band using a new protocol that would encapsulate our data. While this may have limited uses for special cases (such as dealing with IP fragments), a general solution based on inserting additional packets requires a means of authenticating these packets. This is because, presumably, the number of inserted packets is many orders of magnitude less than the number of packets inserted by the attacker. Thus, because we assume that an attacker can insert any packet into the network, the victim can be deluged with fake traceback packets, preventing any information to be gained from the legitimate packets.

## 7.3 The IP Header

Our last source of bits is the IP header. There are several fields in the header that may be exploited for bits, with varying tradeoffs. As shown in Figure 2, we have found 25 bits that might possibly be used.

### 7.3.1 The TOS Field

The type of service field is an 8 bit field in the IP header that is currently used to allow hosts a way to give hints to routers as to what kind of route is important for particular packets (maximized throughput or minimized delay, for example) [1]. This field has been little used in the past, and, in some limited experiments, we have found that setting this field arbitrarily makes no measurable difference in packet delivery. There is a proposed Internet standard [20] that would change the TOS field to a “differentiated services field.” Even the proposed DS field has unused bits, however, there are already other proposed uses for these bits (e.g. [23]).

### 7.3.2 The ID Field

The ID field is a 16 bit field used by IP to permit reconstruction of fragments. Naive tampering with this field breaks fragment reassembly. Since less than 0.25% of all Internet traffic is fragments [26], we think that overloading this field is appropriate. A more in-depth discussion of the issues related to its overloading can be found in Savage’s work [24].

### 7.3.3 The Unused Fragment Flag

There is an unused bit in the fragment flags field that current Internet standards require to be zero. We have found that setting this bit to one has no effect on current implementations, with the exception that when receiving the packet, some systems will think

it is a fragment. The packet is still successfully delivered however, because it looks to those systems as though it is fragment 1 of 1.

### **Our Selection**

We could choose to use up to 25 bits out of the ID, flag, and TOS fields. This would suffice for all of the examples given in Section 6.1.2. The implications of using multiple fields in the IP header simultaneously are modest, since the lost functionality appears to be the union of what would break due to overwriting each field separately. The impact on header checksum calculation is modest, as this can be done in hardware using the standard algorithm.

Of course, the algebraic marking scheme is independent of the choice of bits. The decision of where to put the marking data must be seen as conditional, subject to change as new standards arise.

## **7.4 IPsec**

The interoperability of a traceback scheme with IPsec should be considered. The Encapsulated Security Payload (ESP) [17], which encrypts a datagram for confidentiality, provides no problem for a traceback scheme, as it does not assume anything about the surrounding datagram's headers. The Authentication Header (AH) [16], does present an issue for IPv4. Using the AH, the contents of the surrounding datagram's headers are hashed. Certain header fields are considered mutable (e.g., Fragment Offset), and not included in the hash computation. Unfortunately, the mutable fields in the IPv4 header are unusable for traceback: they either are necessary for basic IP functionality, or reusing them breaks backward compatibility with current IP implementations.

## **7.5 IPv6**

Since IPv6 does not have nearly as many backwards compatibility issues as IPv4, the logical place to put traceback information is a hop-by-hop option in the IPv6 header [10]. However, schemes such as those presented here are still valuable because they use a fixed number of bits per packet thereby avoiding the generation of fragments. Unlike the case in IPv4, we can set the appropriate bit in the Option Type field to indicate that the data in the option is mutable, and should be treated as zero for the purposes of the Authentication Header.

We have not worked out the best way to accommodate IPv6's 128-bit addresses, but note that due to alignment issues, one is likely to select an option length of  $8n + 6$  bytes,  $n \geq 0$ . It would likely be the case that  $0 \leq n \leq 4$ .

# **8 Conclusion and Future Work**

We have presented a new algebraic approach for providing traceback information in IP packets. Our approach is based on mathematical techniques that were first developed for problems related to error correcting codes and machine learning. Though we have

proposed it in the context of a probabilistic packet marking scheme, our algebraic approach could also be applied to an out-of-packet scheme. The resulting scheme would have the desirable property of allowing multiple routers to act on the extra packet while it remains at a small constant size. Our marking schemes have applications for other network management scenarios besides defense against denial of service.

One important open problem is to find better instantiation of the specific methods we have proposed. In particular, a successful approach based on full path tracing would be attractive. More generally, it would be interesting to explore resource and security tradeoffs for the many parametrizations of our schemata. Lower bounds on the size of any marking scheme would be most helpful. It would also be interesting to explore the use of algebraic geometric codes in marking schemes.

## Acknowledgments

We would like to thank David Goldberg and Dan Boneh for valuable discussions. We would also like to thank Dawn Song, Adrian Perrig, Ramarathnam Venkatesan, Glenn Durfee, and the anonymous referees for helpful comments on earlier versions of this paper.

## References

- [1] P. Almquist. Type of service in the internet protocol suite. RFC 1349, July 1992.
- [2] S. Ar, R. J. Lipton, R. Rubinfeld, and M. Sudan. Reconstructing algebraic functions from mixed data. In *33rd Annual Symposium on Foundations of Computer Science*, pages 503–512, Pittsburgh, Pennsylvania, 24–27 Oct. 1992. IEEE.
- [3] S. M. Bellovin. Personal Communications, May 2000.
- [4] S. M. Bellovin. ICMP traceback messages. <http://www.research.att.com/~smb/papers/draft-bellovin-itrace-00.txt>, Mar. 2000.
- [5] E. Berlekamp and L. Welch. Error correction of algebraic block codes. United States Patent 4,490,811, Dec. 86.
- [6] E. R. Berlekamp. *Algebraic Coding Theory*. Aegean Park Press, 1984.
- [7] Bleichenbacher and Nguyen. Noisy polynomial interpolation and noisy chinese remaindering. In *Advances in Cryptology – Eurocrypt 2000*. Springer-Verlag, 2000.
- [8] H. Burch and B. Cheswick. Tracing anonymous packets to their approximate source. In *USENIX LISA 2000*, New Orleans, Louisiana, Dec. 2000. USENIX.
- [9] CERT coordination center denial of service attacks. [http://www.cert.org/tech\\_tips/denial\\_of\\_service.html](http://www.cert.org/tech_tips/denial_of_service.html), Feb. 1999.
- [10] S. Deering and R. Hinden. Internet protocol, version 6 (IPv6) specification. RFC 2460, Dec. 1998.
- [11] D. Dittrich. The “stacheldraht” distributed denial of service attack tool. <http://staff.washington.edu/dittrich/misc/stacheldraht.analysis.txt>, Dec. 1999.
- [12] D. Dittrich. The “Tribe Flood Network” distributed denial of service attack tool. <http://staff.washington.edu/dittrich/misc/tfn.analysis>, Oct. 1999.
- [13] T. Doepfner, P. Klein, and A. Koyfman. Using router stamping to identify the source of IP packets. In *7th ACM Conference on Computer and Communications Security*, Athens, Greece, Nov. 2000. ACM.

- [14] P. Ferguson and D. Senie. Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing. RFC 2267, <http://www.ietf.org/rfc/rfc2267.txt>, Jan. 1998.
- [15] V. Guruswami and M. Sudan. Improved decoding of Reed-Solomon and algebraic-geometric codes. *IEEE Transactions on Information Theory*, 45:1757–1767, 1999.
- [16] S. Kent and R. Atkinson. IP authentication header, Nov. 1998. RFC 2402, <http://www.ietf.org/rfc/rfc2402.txt>.
- [17] S. Kent and R. Atkinson. IP encapsulating security payload (ESP), Nov. 1998. RFC 2406, <http://www.ietf.org/rfc/rfc2406.txt>.
- [18] D. E. Knuth. *The Art of Computer Programming, Vol. 2: Seminumerical Algorithms*. Addison-Wesley, 1998.
- [19] H. Lee and K. Park. On the effectiveness of probabilistic packet marking for ip traceback under denial of service attack. In *IEEE INFOCOM 2001*, Anchorage, Alaska, Apr. 2001. IEEE.
- [20] K. Nichols, S. Blake, F. Baker, and D. Black. Definition of the Differentiated Services field (DS field) in the IPv4 and IPv6 headers. RFC 2474, Dec. 1998.
- [21] V. Olshevsky and M. A. Shokrollahi. A displacement approach to efficient decoding of algebraic-geometric codes. In *31st Annual ACM Symposium on Theory of Computation*, pages 235–244, Atlanta, Georgia, May 1999. ACM.
- [22] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in FORTRAN: The Art of Scientific Computing*. Cambridge University Press, 1992.
- [23] K. Ramakrishnan and S. Floyd. A proposal to add Explicit Congestion Notification (ECN) to IP. RFC 2481, Jan. 1999.
- [24] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Practical network support for IP traceback. In *2000 ACM SIGCOMM Conference*, Aug. 2000.
- [25] D. Song and A. Perrig. Advanced and authenticated marking schemes for IP traceback. Technical Report UCB/CSD-00-1107, University of California, Berkeley, June 2000.
- [26] I. Stoica and H. Zhang. Providing guaranteed services without per flow management. In *ACM SIGCOMM '99*, pages 81–94, Cambridge, MA, 1999.
- [27] M. Sudan. Algorithmic issues in coding theory. In *17th Conference on Foundations of Software Technology and Theoretical Computer Science*, Kharagpur, India, 1997.
- [28] M. Sudan. Decoding of Reed Solomon codes beyond the error-correction bound. *Journal of Complexity*, 13(1):180–193, Mar. 1997.