Probabilistic Packet Marking for Large-Scale IP Traceback

Michael T. Goodrich, *Senior Member, IEEE*

***Abstract*— This article presents an approach to IP traceback based on the probabilistic packet marking paradigm. Our approach, which we call *randomize-and-link*, uses large checksum *cords* to "link" message fragments in a way that is highly scalable, for the checksums serve both as associative addresses and data integrity verifiers. The main advantage of these checksum cords is that they spread the addresses of possible router messages across a spectrum that is too large for the attacker to easily create messages that collide with legitimate messages.**

***Index Terms*— Distributed denial of service, IP, traceback, probabilistic packet marking, checksum cords, associate addresses.**

I. Introduction

A. Modeling the Problem

One of the insidious aspects of distributed denial-of-service attacks is that they maliciously use the strength of routers—to move packets quickly—in that they exploit many different compromised hosts as "zombies" to fire packets at a victim. In order to model DDOS attacks, we consider the attack as propagating in a tree T, where the root of the tree T is the victim, V, and each node in T corresponds to a router X on the Internet that is downstream from an attack host to the victim. From the perspective of V, the tree T is a subtree of a much larger *universal* tree U that consists of the union of all routes to V. (See Figure 1.)

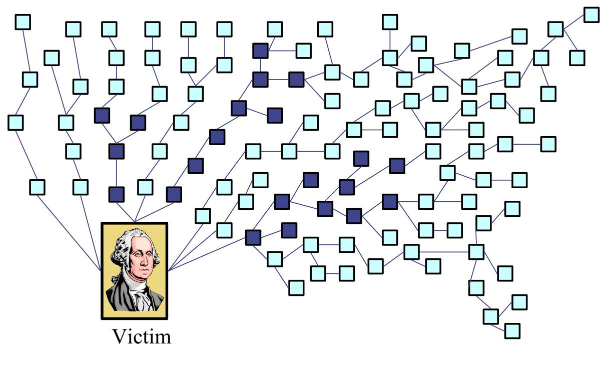

Fig. 1. The universal tree U and attack tree T. The nodes in T are shown darkened.

The goal in the traceback problem is to identify the leaves of T, that is, the routers upstream from the victim closest to attack hosts. We model the attacker as an adversary, A, who can compromise many hosts and use them as "zombies" in a DDOS attack. We allow that A may have knowledge about our traceback algorithms, and that he can even try to design his DDOS attack so as to confuse, break, or delay our algorithms, by "spoofing" the IP headers of attack packets. In this context, we define a *DDOS attack* to consist of a stream of many attacks packets sent from the attack hosts to the victim (in an attempt to overwhelm the victim). We assume that the attacker cannot compromise routers, however. In so doing, A can make it difficult for us to identify some of the routers in the attack tree T. In addition, we allow that A may know the IP addresses of routers in the Internet; hence, he can try to trick us so as to implicate routers not in T.

Dr. Goodrich is a professor in Department of Computer Science, University of California, Irvine, CA 92697-3435. Email: `goodrich(at)ieee.org`.

A major challenge in the IP traceback problem is that there are over 400 million hosts on the Internet. Conservatively assuming that there is a router for every 100–200 hosts on the Internet, we therefore estimate the number of routers (internal nodes) in U to be at least two million. Thus, in practical terms, solving the IP traceback problem amounts to correctly identifying a few thousand of the million internal nodes in U as forming the leaves of the attack tree T. Ideally, we would like to do this identification without requiring any *a priori* knowledge of the universal tree U on the part of the victim V, for such information could be difficult to obtain and maintain.

Finally, we desire solutions to the IP traceback problem that are fast and efficient. We prefer solutions that minimize the amount of additional traffic on the Internet needed to solve the traceback problem or create an infrastructure for solving it. Likewise, we want to allow for incremental adoption by routers in any new infrastructure needed for traceback, and we want to minimize the amount of state that must be maintained by routers. That is, if only a subset U' of routers on the Internet implement our protocol, then we want our traceback algorithm to still work correctly, in this case to identify the leaves of the tree $T \cap U'$. In addition, the computations needed on the part of the victim to identify the leaves of the attack tree T should be fast enough so that V can quickly reconstruct T.

B. Pattern-based Filtering and Hop-by-hop Tracing

In some cases, such as in reflector-based attacks, we can use patterns in the attack packets to filter out DDOS packets at a firewall. Likewise, the approach of *hop-by-hop* tracing, which is also known as *link testing*, uses a pattern-based approach to do traceback of a DOS attack while it is in progress. This is the approach of the automated Pushback mechanism [8], for example, and it is the solution currently supported manually by many router manufacturers. In this approach, a network administrator or his/her agent logs into the routers nearest the victim, and using statistics and pattern analysis, determines the next upstream routers in the attack tree T. The approach is then repeated at the upstream routers for as long as the attack continues. This scheme therefore requires immediate action during the attack, and requires considerable coordination between network administrators (to either communicate directly or setup access points for the agents of partnering administrators). This technique also requires some pattern-based way to separate legitimate packets from attack packets. A similar approach is used by Burch and Cheswick [4] to perform traceback by iteratively flooding from V portions of the Internet to see its effects on V's incoming traffic. Unfortunately, because of their iterative nature, these approaches have limited traceback capabilities in a large-scale DDOS attack.

C. ICMP Messaging

An alternative approach, based on *ICMP messaging* [3], is to have each router X decide, with some probability q (typically, $q = 1/20000$ is mentioned), for each packet P to send an additional ICMP packet to the destination, which identifies X and some content of P. The main idea of this approach is that during a DDOS, a sufficient amount of attack packets will trigger ICMP messages from the routers in the attack tree T so that the victim can identify the leaves of T from these messages. The main drawback of this approach is that it causes additional network traffic even when no DDOS is present. Even so, it is not efficient, for identifying the n leaf nodes in the attack tree T requires, according to the analysis of the coupon collectors problem (e.g., see Motwani and Raghavan [10]), an expected number of nH_n/q packets are needed to arrive at the victim, where H_n is the n-th Harmonic number. For example, if $n = 1000$, then the expected number of attack packets needing to arrive at the victim V before V will have sufficient information to identify the n leaves of T is 138 million.

D. Logging

In addition to the hop-by-hop and ICMP messaging approaches, several researchers have advocated a logging approach to the IP traceback problem. In a logging solution, we either ask routers to log the packets they process or we augment the data packets themselves to contain a full log of all the routers they have encountered on their way to their destinations. Stone [14] and Baba and Matsuda [2] advocate logging of packet information at the routers, and Snoeren *et al.* [12] propose the logging of message digests of packets at the routers. The drawback with these approaches is that they require additional storage at the routers.

E. Probabilistic Packet Marking

An intriguing alternative solution to the IP traceback problem is *probabilistic packet marking*. This traceback approach, which we follow in this paper, can be applied during or after an attack, and it does not require any additional network traffic, router storage, or packet size increase.

Probabilistic packet marking was originally introduced by Savage *et al.* [11]. In this approach to the IP traceback problem, each router X performs, for each packet it processes, an information injection event that occurs with a set probability p (e.g., $p = 1/20$). The information injection involves using b bits in the IP header that are typically not used or changed by routers (they identify the 16-bit IP identification field). They use 5 bits of this field for a hop count, which helps their reconstruction algorithm. The remaining bits are used for the message M_X that the router X wishes to send. If that message is too big, they break it into fragments and use the $b-5$ bits of usable IP header to store a fragment offset and its data fragment. By then including a hash interleaved with the message M_X, the victim can reconstruct M_X from the packets it receives during the DOS attack. Their algorithm is quite interesting, as it introduces the packet marking framework, and does not require *a priori* knowledge of the universal tree U. But their algorithm, unfortunately, is not practical for large distributed denial-of-service attacks. In particular, their algorithm for reconstructing a message M_X from a router at distance d from the victim requires n_d^l checksum tests, where n_d is the number of routers in T at distance d from V and l is the number of fragments messages have been divided into (and this bound generously assumes there are no "noise" packets from the adversary). For example, if $n_d = 30$ and $l = 8$, then the victim has to perform over 650 trillion checksum tests in order to reconstruct each of the 30 messages. Such a computation is, of course, not feasible for the victim, and even if it were, it would introduce many false positives. Moreover, this scheme is easily spoofed by an adversary who knows this algorithm.

Song and Perrig [13] improve the performance of probabilistic packet marking and suggest the use of hash chains for authenticating routers. They also use a 5-bit distance field, but they do not fragment router messages. Instead, they assume the victim knows the universal tree U, and they include a $(b-5)$-bit XOR of hashed message authentication codes (HMACs) from each router X and its downstream neighbor Y. Once a time-released key is revealed, which is a computation performed out-of-band, the victim uses his/her knowledge of U and the revealed keys to determine which routers have marked the given packets. The computation proceeds breadth-first from V, so that each phase requires $n_{d-1}N_d$ HMAC tests, where n_{d-1} denotes the number of routers in T at distance $d-1$ from V and N_d denotes the number of routers in the universal graph U at distance d from V. For example, if $n_{d-1} = 50$ and we conservatively estimate that the number of routers in U at distance d is 100,000, then their algorithm would perform 5 million HMAC tests to determine the routers at distance d in T. Summing over distances $d = 5$ to $d = 25$ (and assuming that near-by tests are faster), implies a total of at least 100 million HMAC tests, which is several orders of magnitude better than a similar reconstruction in the Savage *et al.* approach. Such a computation is still a great effort for the victim, of course, but it is at least feasible. Unfortunately, using an 11-bit HMAC implies that over 45,000 of these tests will be validated at random. Thus, the authentication aspects of their algorithm has scalability issues in addition to the drawback of requiring knowledge of U.

Dean *et al.* [5] introduce an algebraic approach to probabilistic packet marking, where a router X will mark the b reusable bits of a packet with probability p, as in the previous schemes, but the marking information is the value of a linear polynomial with X's identity as its leading coefficient. Any subsequent router Y not initiating a similar computation nevertheless changes the b-bits by performing an iteration of Horner's rule to create an evaluation of a new polynomial having the IP addresses of the routers on the path from X to Y as its coefficients. Thus, in order to reconstruct each path in the attack tree, the victim must perform polynomial interpolation with noise on the packets it has received. Dean *et al.* identify 25 bits in the IP header that can be used for marking, namely, 16 bits from the ID field used in fragmentation (which is used only 0.25% of the time), 8 type-of-service bits, and one flag bit (which is also used in fragmentation). Unfortunately, their interpolate-with-noise algorithms are complex and slow for large distributed denial-of-service attacks.

Lee *et al.* [9] show how to add statistical analysis to an existing probabilistic packet marking scheme, so that, in addition to identifying the routers that are downstream from attack hosts, they can estimate the average traffic rate for each edge in the attack tree. Their scheme appears to work with any probabilistic packet marking scheme, so combining it with our approach should allow for traffic analysis of larger attack trees.

F. Our Results

In this paper we introduce a novel approach to probabilistic packet marking, which we call the *randomize-and-link* approach, that greatly improves the practicality and security of probabilistic packet marking. The main idea of our approach is to have each router X fragment its message M_X into several words and include in the b reusable bits such a word fragment at random together with a large checksum *cord* on the entire message M_X. For example, if $b = 25$, we may wish to include 14 bits of a checksum cord in every marked packet. Such an approach to packet marking may at first seem counter-intuitive, for we are apparently wasting a large amount of "real estate" in the precious b bits. But the checksum cords make the reconstruction algorithm much more efficient. The checksum cords serve both as associative addresses for the router messages and also as partial integrity validators. They also spread the spectrum of possible messages across a large domain, which significantly reduces the ability of the adversary to interject false messages that collide with legitimate ones. In addition, by including reasonably-large HMAC information in the message M_X, we can achieve unpredictability for these checksum cords, which makes the adversary's job harder, while also providing moderate-to-strong authentication of the routers in the attack tree T. Moreover, our algorithms do not require any knowledge of the universal tree U, and we avoid the requirement of having routers sign individual setup messages by employing authenticated dictionaries [1], [7] for the out-of-band validations. In addition, our scheme can be deployed *incrementally*—not all routers need be using our scheme in order for it to work.

II. Efficient Packet Marking

Let b denote the number of bits in the IP header that we can safely use to encode information from a router. For example, we may wish to use $b = 25$, as advocated by Dean *et al.* [5]. Indeed, we will use $b = 25$ as a running example throughout most of this paper. Still, even if one does not use the 8 type-of-service bits (which are being advocated for differentiated services), we would still have $b = 17$ (and we give some examples using this value for b as well). In either case, we may sometimes upset packet fragmentation, but the frequency of fragmentation is arguably below typical packet loss rates [5].

A. High-Level View

Our scheme for sending to V the message M_X from each router X in the attack tree is based on using a technique that we call *randomize-and-link*. The main idea of this technique is to perform the following transformation on M_X:

1) Pad M_X as needed to make $|M_X|$ a multiple of l, which is a parameter in our algorithm.
2) Compute a reasonably large (and statistically random) checksum $C = C(M_X)$ on the sequence M_X. The checksum $C(M_X)$ should utilize randomness in itself or M_X, so that $C(M_X)$ is statistically random (like a random hash function) and unpredictable to the adversary.
3) Break M_X into a sequence W of non-overlapping word fragments $M_0, M_1, M_2, \ldots, M_{l-1}$.
4) Create a collection of blocks, which are used to overwrite the b bits, so $b_i = [i, C, M_i]$.

Thus, the block consists of an index, checksum cord, and a message fragment. (See Figure 2.)

We use these b_i blocks to transmit the message M_X to the victim V. These pieces are not sent in any particular order, however. We call $C = C(M_X)$ the *cord* for M_X, as it will be used as both an associative address for M_X and a checksum to "link" all the pieces of M_X back together. Moreover, since the cord C is statistically random and unpredictable to the attacker, he cannot easily create false cords that would confuse the reconstruction algorithm. This reconstruction algorithm is therefore quite simple—given a collection of b_i's with the same cord C, a victim simply tries all possible ways of putting the b_i's back together in the right order, using the checksum property of the cord C to eliminate unintended sequences. Once the victim V has a valid sequence of b_i's correctly constructed in order, V will have built the message M_X. We give the details below. (See also Figure 3.)

B. Randomize-and-Link Transmission

As mentioned above, we assume that the IP header allows the reuse of some of its bits for the purpose of information marking by routers. We partition the b reusable bits in the IP header as follows:

- $\lceil \log l \rceil$ bits for the fragment index i
- c bits for the cord, which serves both as an associative address and as a checksum
- $h = b - c - \lceil \log l \rceil$ bits for the data word M_i.

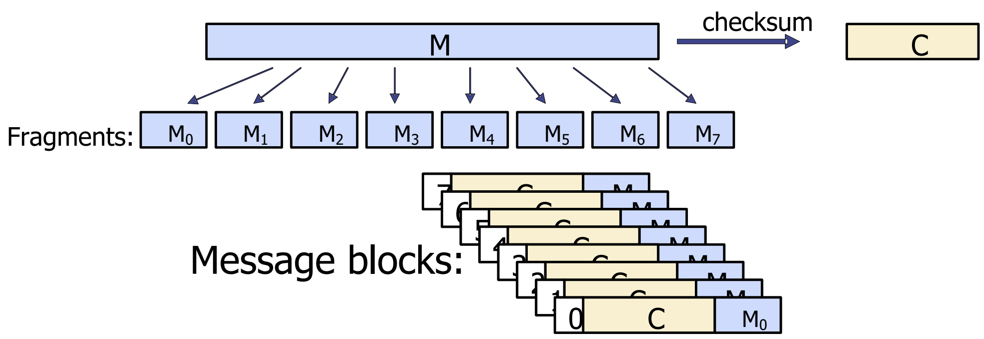

Fig. 2. The message blocks in our packet marking scheme.

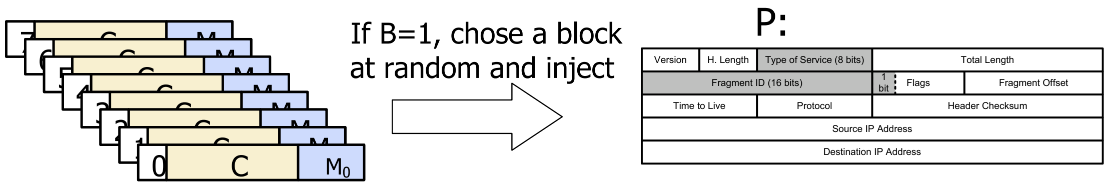

Fig. 3. The block injection process. Potentially-reusable bits are shown shaded.

For example, if $b = 25$ and $l = 8$, then we could use $\lceil \log l \rceil = 3$ bits for the index i, $c = 15$ bits for the checksum C, and $h = 7$ bits for each data word M_i.

We assume that either the function $C()$ or M_X itself contain a sufficient randomness so that the checksum value $C(M_X)$ is statistically random and unpredictable to the adversary. That is, it is as unlikely as a random hash function with similar output size for $C(M_X) = C(M_Y)$ for two different router messages M_X and M_Y. In particular, we want $C(M_X)$ to be unpredictable to an adversary who knows only the value of X (we assume the adversary does not know all of M_X). For example, if M_X does not contain sufficient randomness in itself, we could pad M_X with a random nonce.

We pad M_X to have a size that is a multiple of l and we compute the c-bit checksum cord $C = C(M_X)$ on M_X, and we break M_X into a sequence W of l words $M_0, M_1, M_2, \ldots, M_{l-1}$ of length h bits each. We define a set of l blocks $b_0, b_1, \ldots, b_{l-1}$ so that $b_i = [i, C, M_i]$. Note that the cord C is included in every block b_i. Indeed, it is the inclusion of the cord C that links the blocks b_i together, as it makes C an associative address for the blocks.

C. Packet Marking

Once we have the blocks $b_0, b_1, \ldots, b_{l-1}$ defined for a message M_X, we proceed with probabilistic packet marking in the natural way. Namely, we define a probability parameter p (e.g., $p = 1/20$). With each packet that X receives, we "flip a coin" with probability p. If this coin comes up "tails" (an event that occurs with probability $1 - p$), then X simply forwards the packet on to its destination as usual. Otherwise, if the coin comes up "heads," then X chooses one of its blocks b_i at random, inserts b_i into the reusable bits of this packet (updating the header checksum as needed), and forwards this revised packet on to its destination.

This packet marking process continues until we choose, for timeliness reasons, to change the message M_X. At such a time that we wish to change to a new M_X, the router X repeats that above computation for the b_i blocks for the new message. The router then repeats the probabilistic packet marking for this new set of blocks, until we decide yet another change is needed. Thus, we keep very little state at a router in order to implement the randomize-and-link packet marking scheme. A router doesn't even need to store the blocks $b_0, b_1, \ldots, b_{l-1}$, so long as it has a fast way of generating a b_i at random. Moreover, note that the computational overhead per packet is very small. In the default case, when the "coin flip" is tails, the router's work is the same as if it were doing no packet marking at all; hence, this scheme can be deployed incrementally.

D. Message Reconstruction

The message reconstruction algorithm is based on a simple combinatorial process. Given a set of packets received at the victim, we sort their b-bit blocks lexicographically by their (C, i, M_i) values, and remove duplicates (interpreting values according to the same format we used to store blocks in the IP header). This sorting can be done, for example by a radix sort. Thus, we have, for each distinct cord C, all the distinct blocks for this cord ordered by their i-index. We let $P_{C,i}$ denote the set of distinct packets that have cord C and fragment index i. We then try all combinatorial combinations of the blocks in $P_{C,0} \cdot P_{C,1} \cdots P_{C,l-1}$, computing a checksum for each. We keep only those combinations that have a checksum equal to the cord C. That is, we accept these strings as being strong candidates as having been sent from the routers (although we must recognize that some of these may have been sent by the attacker). (See Figure 4.)

The total running time, then, for this reconstruction algorithm at the victim is proportional to the following quantity:

$$N + \sum_{C} \prod_{i=0}^{l} N_{C,i},$$

where N is the total number of packets and $N_{C,i}$ is the number

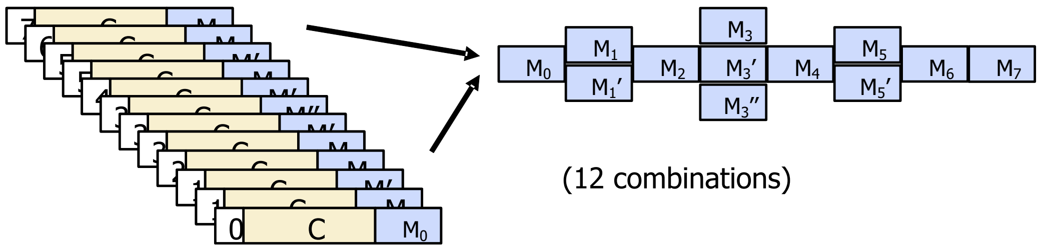

Fig. 4. The message-fragment combination process.

of distinct packets from this set with cord C and fragment index i.

E. Two-Phase Fragmentation

In the above discussion, we argued how fragmenting a message into small blocks indexed (that is, linked) by a large statistically-random checksum cord can be an effective means for sending a message to the victim that is longer than b bits. In particular, fragmenting a message into two, four, or eight word fragments can be an efficient way to send a moderate-sized message to the victim (say, on the order of 48 to 96 bits). Unfortunately, if we have a larger-sized message (say, on the order of 128 or 192 bits), eight fragments may not be sufficient to send the message and still utilize a large checksum cord (which is needed for both security and message reconstruction). We can iterate our randomize-and-link approach, however, to send larger messages. We begin as in our previous method. We take the message M and subdivide it into l words, $M_0, M_1, \ldots, M_l$. This subdivision should be done in such as way as to preserve in each word M_i the same degree of randomness as is present globally in the message M. Still, in many cases where we want M to be reasonably large, we may observe that each word M_i is too big to be transported with high confidence in a single data block. So we further subdivide each word M_i into m subwords $M_{i,0}, M_{i,1}, \ldots, M_{i,m}$. Given the value m and the size of the subwords, we determine the number, c_1, of checksum bits that we can devote to sending the subwords in the first round (given our fixed size of b bits per block). We devote $b - c_1 - \lceil \log m \rceil$ bits to the data in each subword $M_{i,j}$. Thus, we can compose subwords to form bigger blocks of $m(b - c_1 - \lceil \log m \rceil)$ bits. In order for these big blocks to have the same security as the smaller blocks, we should devote $c_2 = c_1 - \lceil \log m \rceil$ bits to a random checksum cord for each of them, just as we did in our single-phase approach. This factor is due to the fact that the probability of collision between two distinct packets in the first round is $1/m2^{c_1}$ and this probability in the second round is $1/lm2^{c_2}$, since every round-two word was comprised of m round-one subwords. In addition, we must also devote $\lceil \log l \rceil$ bits to a fragment number of each index i. So, for each word M_i we compute a c_2-bit checksum cord we wish to use in order to achieve high confidence of message transmission for each word. (See Figure 5.)

Data transmission in the two-level scheme is as in the one-level scheme, except that now when a router decides to interject a message into a packet it chooses one of its many subwords, $M_{i,j}$ at random and interjects this. Reconstruction of the message, of course, proceeds in two phases. In the first phase we reconstruct all the candidate words M_i and in the second phase we reconstruct all the candidate messages. Thus, the running time for message reconstruction in the two-phase scheme is proportional to the following:

$$N + \sum_C \prod_{i=0}^{m} M_{C,i} + \sum_C \prod_{i=0}^{l} N_{C,i},$$

where N is the total number of packets the victim is using for reconstruction, $M_{C,i}$ is the number of distinct phase-one blocks from this set with cord C and fragment index i, and $N_{C,i}$ is the number of distinct phase-two blocks with cord C and fragment index i. In the analysis section that follows, we show that these quantities can be quite reasonable, provided that there are a sufficient number of bits devoted to the checksum cords.

III. Analysis

We begin our analysis by estimating the number of packets that are needed for traceback in our single-phase method. Let n denote the number of leaf routers in the attack tree T, and recall that l is the number of words in the message M_X each router X wishes to transmit to the victim V. Thus, the victim wishes to receive nl distinct packets if we are to reconstruct the messages from all the leaf routers in T.

Let p denote the probability that a router inserts its information into a packet that it is routing. So, $p(1-p)^{d-1}$ is the probability that a packet is marked and arrives unchanged from a router that is d hops away from V. If we conservatively assume that all routers have their packets successfully delivered with probability at least that of the farthest routers, then we can safely estimate that the information from some leaf router in the attack tree will be contained in a packet received by the victim with probability at least $p(1-p)^{d-1}$, where d is the maximum hop-distance for any such router.

Since every router must successfully send l different blocks for all of its information to arrive at the victim, the expected number of packets that must be received before all fragments have been received is an instance of the coupon collectors problem [10], where the number of "coupons" is nl and the probability of receiving a marked packet is at least $p(1-p)^{d-1}$. This observation implies that the expected number of packets that must arrive at the victim before it can identify the n leaf routers of T is at most

$$\frac{nlH_{nl}}{p(1-p)^{d-1}},$$

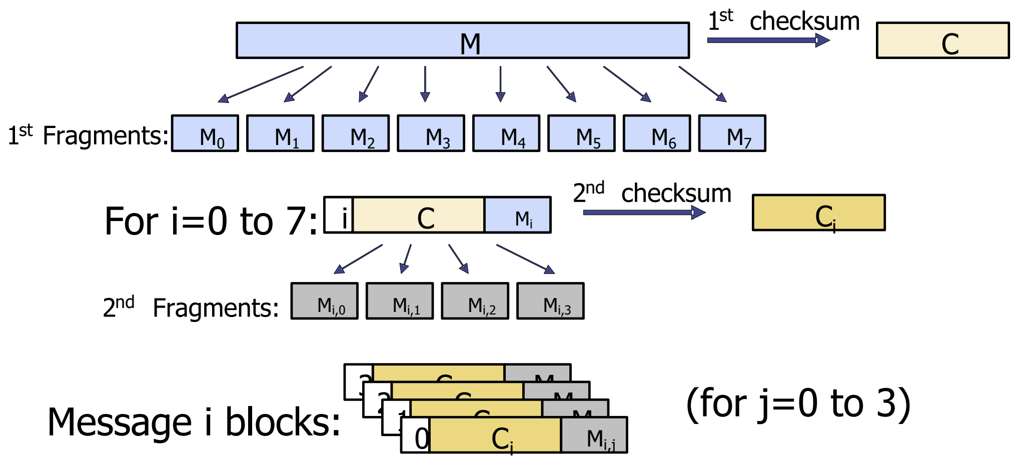

Fig. 5. The two-phase fragmentation scheme.

where H_n denotes the n-th Harmonic number. Using a well-known inequality for H_n,

$$H_n < \ln n + \gamma + \frac{1}{2n},$$

where $\gamma = 0.5772156649...$ is Euler's constant. Thus, the expected number of packets that must arrive at V before it can perform a complete traceback of n routers using our scheme is at most

$$\frac{nl\ln(nl) + \gamma nl + 1}{p(1-p)^{d-1}}.$$

For example, if $p = 1/20$, $d = 10$, $n = 1000$, and $l = 8$, then the expected number of needed packets to do reconstruction of all the leaf router messages is 76516, not considering the minimum packet marking probability, $p(1-p)^{d-1}$. Dividing this expectation by the minimum packet marking probability in this case implies that the expected total number of packets needed by the victim to do complete reconstruction of all messages is at most 2428118. Note that there are only nl distinct packets that come from the leaf routers in the attack tree. The challenge for the attacker is that he cannot predict the checksum cords; hence, it is unlikely that the packets that get routed to the victim without marking (which occurs with probability $(1-p)^d$) will be confused for marked packets.

The analysis of the two-phase version of the randomize-and-link algorithm is similar to that given above for the single-phase version. The main difference is that in the two-phase algorithm we wish to receive, from each router, l words subdivided into m subwords. That is, we wish to receive lm packets from each leaf router in T. Thus, the expected number of packets we have to receive in order to do complete traceback is

$$\frac{nlmH_{nlm}}{p(1-p)^{d-1}}.$$

Let us address next the expected running time needed to reconstruct all the messages received by the victim. We give the analysis first for our single-phase algorithm, and we then explain the slight differences for our two-phase algorithm.

The important observation in analyzing the expected running time of the message reconstruction algorithm is that, since the checksum cords are statistically random, we can view the mapping of messages to checksum indices as a random hash function. Thus, the number of collisions among legitimate messages should be small. Of course, the adversary might construct lots of fake messages and then construct lots of collisions with these messages, but let us ignore this possibility for the time being (we will revisit this possibility shortly).

We begin with the reconstruction algorithm for our single-phase scheme. Let N denote the number of distinct packets the victim has received and let n denote the number of routers in the attack tree. Since there are l pieces to each message and each one has a c-bit checksum cord that is statistically random, the probability that two random packets have the same fragment index i and cord C is $1/(l2^c)$. In addition, the probability that two router-sent packets have the same fragment index i and cord C is at most $1/2^c$, since any router wishing to send a message will send a packet with each i-index for the checksum cord C it is using. Thus, the expected number of packets with the same fragment index i and cord C is at most $\lceil N/(l2^c) + n/2^c \rceil$. For any given cord C, then, the running time for computing all combinatorial combinations of blocks with this cord is proportional to $Z = Z_0 Z_1 \cdots Z_{l-1}$, where Z_i is a random variable corresponding to the number of packets with fragment index i (for this checksum cord C). Since these Z_i's refer to different fragment indices, they are independent; hence, the expected value of their product is equal to the product of their expected values. That is, the expected running time for checking all the combinations for a given checksum cord C is $\lceil (N + ln)/(l2^c) \rceil^l$. Summing this expectation over the at most N possible cord values, we see that the expected number of checksum tests in the message reconstruction phase is

$$\left(\frac{N}{l}\right) \left\lceil \frac{N + ln}{l2^c} \right\rceil^l,$$

where the N/l bound comes from the fact that we have to have at least l different fragments for a given checksum cord before we will have to perform an actual checksum test. So, for example, if $N = 80000$, $l = 8$, $n = 1000$, and $c = 14$, then the expected number of checksum tests the victim must make is only 10,000.

The adversary has little advantage in our two-phase algorithm, as well, although the total number of packets needed in this case is somewhat greater than in the single-phase approach. Specifically, in the two-phase algorithm, each router sends a total of lm fragments. The victim first assembles these as m-length subwords, and then assembles the l words produced from this reconstruction. If the first phase uses c_1-bit checksum cords and the second phase uses c_2-bit checksum cords, then the expected number of checksum tests is

$$\left(\frac{N}{m}\right)\left\lceil\frac{N+nlm}{m2^{c_1}}\right\rceil^m + \left(\frac{N}{lm}\right)\left\lceil\frac{N+nlm}{lm2^{c_2}}\right\rceil^l .$$

The arguments justifying this bound are similar to those above, but applied twice. So, for example, if $N = 80000$, $l = 8$, $m = 4$, $n = 250$, $c_1 = 14$, and $c_2 = 12$, then the expected number of checksum tests the victim must make is 10,000.

Of course, the adversary may deliberately send false messages that have valid checksum cords according by our scheme. But the number of such messages is limited, for the adversary must be limited to the same coupon collector bounds as the legitimate routers in T. To estimate the number of such false messages, let us conservatively assume that the probability that a packet arrives unchanged from the adversary is equal to the probability that the victim receives a packet marked by a router. Thus, the maximum number of false messages the adversary can send is bounded by n, the number of legitimate routers in the attack tree. Still, the adversary may not only try to send us false message with valid checksum cords. He may also send lots of extra packets that have checksum cords that deliberately collide with each other, so as to make us do extra wasteful work trying in vain to find a combination of these word fragments that have a checksum equal to this cord. Fortunately, as we show below, we can apply a probabilistic packet filtering strategy to our algorithm that significantly limits the amount of extra work the adversary can force us to do in combining colliding word fragments.

We can derive a high-probability upper bound on the running time of the message reconstruction algorithm, which is useful for identifying improbably large numbers of collisions that are most likely deliberately sent by the adversary in an attempt to slow down our traceback algorithm. Armed with this high-probability bound, we can safely discard packets that define an improbably large number of collisions in reconstructing the blocks for a specific cord C. Let us fix a checksum cord C, and let $Y = Z_0 + Z_1 + \cdots + Z_{l-1}$, where Z_i denotes the number of distinct blocks with fragment index i and cord C. We will bound Y and thereby derive a bound on $Z = Z_0 Z_1 \cdots Z_{l-1}$. We will utilize the following Chernoff bound (e.g., see Motwani and Raghavan [10]):

Theorem 1 ((Chernoff Bound Theorem)): Let Y be the sum of independent indicator (0/1) random variables, and let μ denote the expected value of Y. Then,

$$\Pr(Y > (1+\delta)\mu) < \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^\mu .$$

Corollary 2: Let Y be the sum of independent indicator (0/1) random variables, and let μ denote the expected value of Y. For any $k > \mu$,

$$\Pr(Y > k) < \frac{e^{k-\mu}\mu^k}{k^k}.$$

Returning to our analysis, then, note that Y can be defined as the sum of independent indicator random variables and that the expected value of Y is $\mathbf{E}(Z_0) + \mathbf{E}(Z_1) + \cdots + \mathbf{E}(Z_{l-1})$. Thus, in the single-phase algorithm, $\mathbf{E}(Y)$ is $lN/(l2^c) + ln/2^c = (N + ln)/2^c$. Thus, we have:

Lemma 3: Suppose $N \leq l2^c$. Then, for any integer $x \geq 2$,

$$\Pr(Y > lx) < \frac{e^{xl-(N+ln)/2^c}}{x^{xl} l^{l(x-1)}};$$

hence,

$$\Pr(Z > x^l) < \frac{e^{xl-(N+ln)/2^c}}{x^{xl} l^{l(x-1)}}.$$

Thus, we can bound with this same probability that $Y > lx$ the odds that $Z > x^l$. So, for example, if $l = 8$, $c = 15$, $x = 3$, $n = 2^{10}$, and $N = 2^{18}$, then the probability that $Y > 24$ is less than $1/2^{48}$, which is a very small number. Thus, in this example, we may safely discard the packets for any index C that have more than 24 packets. That is, we may safely discard any subproblems that would cause us to perform more than $3^8 = 6561$ checksum tests. With high probability, such a subproblem will not occur at random, so it most likely was sent to us by the adversary in an attempt to make us do extra unnecessary work in our single-phase traceback algorithm. A similar analysis can be done for the two-phase algorithm.

Frag.	Number, c, of Checksum Bits						
scheme	9	10	11	12	13	14	15
2	56	52	48	44	40	36	32
3	104	96	88	80	72	64	56
$2+2$	188	168	148	128	108	88	68
$3+2$	380	344	308	272	236	200	164
$3+3$	760	688	616	544	472	400	328

Fig. 6. Message sizes for the given fragmentation schemes and checksum lengths, assuming $b = 25$ is the bit length of individual blocks. A fragment scheme identified by "x" indicates a scheme with x number of bits used to index fragments. A fragment scheme identified by "$x + y$" indicates a two-phase fragmentation scheme where the first round uses x bits for the fragment index and the second round uses y bits for the fragment index. We index the checksum needs in the two-phase schemes by c_1, since we can set $c_2 = c_1 - \lceil \log l \rceil$.

A. Trading-Off Message and Cord Size

As we observed above, increasing the checksum size leads to higher security. A large checksum makes the space of possible messages addresses large, which in turn makes it more difficult for the adversary to interject false messages that collide with legitimate ones. Of course, this increased security has a cost. Namely, as the checksum becomes larger, the bits left over for the message must go down. Even so, there are still several strong choices for checksum lengths and fragmentation schemes that allow for message sizes long enough to do authenticated IP traceback. We show in Figure 6 the maximum message size for various randomize-and-link fragmentation schemes, assuming $b = 25$, and we show similar information for $b = 17$ in Figure 7.

Frag. scheme	Number, c, of Checksum Bits 8	9	10	11
3	48	40	32	24
$2+2$	80	60	40	20
$3+2$	160	124	88	52
$3+3$	320	248	176	104

Fig. 7. Message sizes for the given fragmentation schemes and checksum lengths, assuming $b = 17$ is the bit length of individual blocks.

Frag. scheme	Number, n, of Routers in Attack Tree 50	100	250	500	1000
2	1176	2628	7486	16357	35486
3	2628	5810	16357	35486	76516
$2+2$	5810	12729	35486	76516	164122
$3+1$	5810	12729	35486	76516	164122
$2+3$	12729	27675	76516	164122	350424
$3+2$	12729	27675	76516	164122	350424
$3+3$	27675	59785	164122	350424	745208

Fig. 8. Expected upper bounds on N, the number of packets that need to be received for various fragmentation schemes and number of routers, n, in the attack tree. The volumes given are the expected number needed to cover all the routers. To convert the presented numbers to an expectation that factors in the marking probability, the above values should be divided by $p(1-p)^{d-1}$.

B. Sufficient Packet Volume

We have described the randomize-and-link strategy in a general way, so as to allow for several possible message sizes. But we should also recognize that reconstructing large messages requires more packets. Moreover, the number of needed packets also increases with the number of routers in the attack tree. In order to keep the reconstruction algorithm fast, we prefer that expected number of collisions between a given packet and any other packet be less than 2. The randomize-and-link algorithm will still work for higher expectations, but it is most efficient when the expected collision size is less than 2. Thus, we have worked out the needed packet volume and checksum bit-length for various randomize-and-link fragmentation schemes under various numbers of routers in the attack tree. We provide this information in Figures 8 and 9.

We have performed an experimental simulation of our scheme to test empirically the average number of packets that must be received by a victim before it can identify all the leaf routers in an attack tree T. We used a complete binary tree for T and assumed that packets were being generated at attack hosts uniformly at random. Experiments were performed to test the average number of packets needed by a victim to identify all of 1000 attack hosts that were 10 hops away, for varying values of the marking probability p. Average packet frequencies are reported in Figure 10 for the total number of packets and the number of unmarked packets. Note that, for high probabilities, most packets are marked, but not by leaf nodes in T, while, for low probabilities, most packets are unmarked. Fortunately, the results show that as long as the probability is roughly equal to the inverse of the distance to the attack hosts, the number of total packets needed to identify all the leaves of T is reasonable.

Frag. scheme	Number, n, of leaf routers in T 50	100	250	500	1000	2000
2	8	9	10	11	13	14
3	8	9	10	12	13	14
$2+2$	10	11	13	14	15	16
$3+1$	9	10	12	13	14	15
$2+3$	11	12	14	15	16	17
$3+2$	10	11	13	14	15	16
$3+3$	11	12	14	15	16	17

Fig. 9. The checksum sizes needed for the different randomize-and-link fragmentation schemes, assuming various values of the number of leaf routers, n, in the attack tree. The checksum sizes given are the number needed to force the expected collision size to be less than 2. For the two-phase schemes, the checksum size is given for the first round, since its checksum needs are higher.

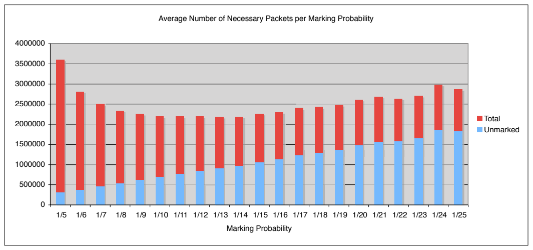

Fig. 10. Average packet frequencies needed to identify 1000 leaves in a binary attack tree T with depth 10, for various marking probabilities. Each frequency is averaged over 100 runs of the experiment.

IV. Router Identification and Authentication

In this section, we discuss how our packet marking algorithm could fit into a complete IP traceback scheme. This discussion is not meant to be exhaustive, however, as the randomize-and-link approach only specifies the length of the information that is sent from the routers in an attack tree to the victim. The main point we want to make in the following discussion, then, is to show how the randomize-and-link strategy can be used to create a traceback method that scales to thousands of routers, that does not require the victim know the universal tree U, and that can authenticate routers without requiring them individually to sign setup messages.

As mentioned above, our general approach is based on the existence of a message M_X that a router X will transmit so that it includes X's identity. That is, M_X includes X's 32-bit IP address. Additionally, if we want to learn edges of the attack tree T, as opposed to simply learning the names of all the leaf routers in T, we can optionally have each router X also include in M_X the name of its downstream router in T, which would add another 32 bits to M_X. This identity and optional topology information, of course, does not provide any additional randomness or authentication information to M_X, as is needed by the security needs of a good randomize-and-link traceback scheme. Fortunately, there are several ways that we can simultaneously add randomness and authentication to M_X. Before we describe one of these ways, however, let us briefly review the main cryptographic tool that this method is based on.

A. Authenticated Dictionaries

One possible authentication scheme utilizes an authenticated dictionary for some portions of the task of authenticating routers. An authenticated dictionary [1], [7] consists of a trusted source and many untrusted directories. The trusted source produces and maintains a dictionary database, D, of objects, stored as key-value pairs, (k, o), while the directories answer key-value queries for D on behalf of client users. In requesting a key-value query, a client provides a key k and asks the directory to return the object o in D that has k as its key. If there is such an object, then the directory returns it. If, on the other hand, there is no object with key k in D, then the directory returns a special "no-such-object" value. In either case, in addition to the answer a directory gives, a directory also provides a cryptographic proof of that answer, which validates (subject to standard cryptographic assumptions) that the answer is current and is as accurate as if it had come directly from the source. There is a small (usually logarithmic) overhead incurred for this cryptographic proof, but it allows the source to be offline during the request. In addition, by deploying many directories widely dispersed in the network, using an authenticated dictionary allows us to reduce response latency and the effectiveness of a denial-of-service attack on the authenticated dictionary itself (for such an attack would have to target all of the directories simultaneously). We describe below how authenticated dictionaries can be used in various traceback schemes to allow for strong authentication of routers without requiring them to sign any setup messages individually.

B. HMAC with Individual Key Exposure

One way to do router authentication is to determine, for each router X, a sequence of secret keys $K_{X,0}, K_{X,1}, \ldots$. Then, with a message M_X intended for a victim V, the router X includes a hashed message authentication code (HMAC) of $h(V||K_{X,t})$, where h is a one-way cryptographic hash function and t is a time-quantum counter that is of sufficient granularity that we can assume rough synchronization between routers and the victim. We include V in the HMAC so as to reduce the possibility of a replay attack. We reveal the key $K_{X,t}$ for each router X in time quantum $t + 2$. The revelation is done using an authenticated dictionary for each autonomous system (AS), whose source is the administrator of the AS. Indeed, we assume this administrator distributed the secret keys to his or her routers in the first place. To determine the amount of extra space this scheme adds to the message M_X, note that we did not include the packet's source address in the HMAC, as is done, for example, in a previous scheme [13], for this value is set by the adversary. Thus, there is no risk of a birthday attack for our hash function h, as its data is fixed for V and the current time quantum t. This scheme will therefore add 32, 48, or 64 bits to M_X depending on whether we want fair, moderate, or strong authentication of X.

a) HMAC with Messaged Key Exposure: An alternate scheme to the previous authentication method is to utilize a sequence of secret keys, $K_{X,0}, K_{X,1}, \ldots$, as before, but include $K_{X,t-2}$ in M_X during time quantum t. In this way, the router X itself reveals the secret key used in the HMAC. In this case, we should create the sequence of keys themselves as a *hash chain* using a one-way cryptographic hash function, g, so that $K_{X,i} = g(K_{X,i+1})$. Then we need only store $K_{X,0}$ in the authenticated dictionary for X's autonomous system. For given any revealed $K_{X,t}$, the victim can determine the authenticity of this key by performing t applications of the function g. Thus, this approach reduces the work required of the AS administrator, so that rather than revealing keys with each time quantum, the administrator now just needs to reveal the base of each router's hash chain. The trade-off is that we now are including more information in M_X. Namely, we are adding 64, 96, or 128 bits to M_X, depending on whether we want fair, moderate, or strong authentication of X. In addition, since the keys are determined through a hash chain, we now require the victim to perform t hash computations for every router in the attack tree. This effort can be significant if the number of routers in T is over 1000.

V. Discussion and Conclusion

We have presented a new approach to IP traceback based on the probabilistic packet marking paradigm. Our approach, which we call randomize-and-link, uses large checksum cords to link message fragments in a way that is highly scalable, for the cords serve both as associative addresses and data integrity verifiers. For example, with a 12-bit checksum cord we can use a single-phase randomize-and-link scheme to produce an 80-bit message that contains a router's 32-bit IP address and a 48-bit combination HMAC. Such a scheme would allow for fast and efficient message reconstruction for up to 500 routers in the attack tree T. If we wish to traceback efficiently attacks that are targeting a victim through a larger attack tree, we could use a 16-bit initial checksum cord in a two-phase randomize-and-link strategy (using 8 subwords in phase one and 4 words in phase two) that produces a 128-bit message. Such a message could contain a router X's IP address, the IP address of the downstream neighbor of X, and a 64-bit HMAC (collective or individual). Or such a message could contain X's IP address, a 48-bit HMAC, and a 48-bit key revelation. In either case, using a 16-bit checksum cord with a two-phase scheme producing a 128-bit message would allow for fast and efficient traceback for attack trees of size up to 2000 routers. In general, our methods do not require that a victim know the topology of the universal tree U, we do not require that routers sign any setup messages individually, and we allow for incremental adoption (for the default router action is to process packets in the same way as a non-participating router).

Acknowledgments

We thank Maithili Narasimha, Dawn Song, and Gene Tsudik for helpful discussions related to this paper. A preliminary version of this paper was announced in [6]. This work was supported by DARPA grant F30602-00-2-0509 and NSF grants CCR-0225642, CCR-0311720, and CCR-0312760.

References

[1] A. Anagnostopoulos, M. T. Goodrich, and R. Tamassia. Persistent authenticated dictionaries and their applications. In *Proc. Information Security Conference (ISC 2001)*, volume 2200 of *LNCS*, pages 379–393. Springer-Verlag, 2001.

[2] T. Baba and S. Matsuda. Tracing network attacks to their sources. *IEEE Internet Computing*, 6(2):20–26, 2002.

[3] S. M. Bellovin. ICMP traceback messages. In *Work in Progress, Internet Draft draft-bellovin-itrace-00.txt*, March 2000.

[4] H. Burch and B. Cheswick. Tracing anonymous packets to their approximate source. In *Usenix LISA (New Orleans) Conference*, pages 313–322, 2000.

[5] D. Dean, M. Franklin, and A. Stubblefield. An algebraic approach to IP traceback. In *Network and Distributed System Security Symposium (NDSS)*, pages 3–12, 2001.

[6] M. T. Goodrich. Efficient packet marking for large-scale IP traceback. In *9th ACM Conf. on Computer and Communications Security (CCS)*, pages 117–126, 2002.

[7] M. T. Goodrich, R. Tamassia, and A. Schwerin. Implementation of an authenticated dictionary with skip lists and commutative hashing. In *Proc. 2001 DARPA Information Survivability Conference and Exposition*, volume 2, pages 68–82, 2001.

[8] J. Ioannidis and S. M. Bellovin. Implementing pushback: Router-based defense against DDoS attacks. In *Proceedings of Network and Distributed System Security Symposium*. The Internet Society, 2002.

[9] T. K. T. Law, D. K. Y. Yau, and J. C. S. Lui. You can run, but you can't hide: An effective statistical methodology to trace back ddos attackers. *IEEE Transactions on Parallel and Distributed Systems*, 16(9):799–813, 2005.

[10] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, New York, NY, 1995.

[11] S. Savage, D. Wetherall, A. R. Karlin, and T. Anderson. Practical network support for IP traceback. In *SIGCOMM*, pages 295–306, 2000.

[12] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, S. T. Kent, and W. T. Strayer. Hash-based IP traceback. In *Proc. of the ACM SIGCOMM 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, 2001.

[13] D. Song and A. Perrig. Advanced and authenticated marking schemes for IP traceback. In *IEEE Infocomm*, 2001.

[14] R. Stone. Centertrack: An IP overlay network for tracking DoS floods. In *Proc. of 9th USENIX Security Symposium*, August 2000.

Michael T. Goodrich received his PhD in Computer Sciences from Purdue University in 1987, and is a professor of computer science at UC-Irvine, where he also serves as director of the Center of Cyber-Security and Privacy. Dr. Goodrich's research is directed at the design of high performance algorithms and data structures for solving large-scale problems motivated from information assurance and security, the Internet, information visualization, and geometric computing. He has pioneered and led research on efficient parallel and distributed solutions to a number of fundamental problems, including sorting, convex hull construction, segment intersection reporting, fixed-dimensional linear programming, polygon triangulation, Voronoi diagram construction, and data authentication.