A New IP Traceback System Against Distributed Denial-of-Service Attacks

Rafael P. Laufer[1], Pedro B. Velloso[1,2], Daniel de O. Cunha[1],
Igor M. Moraes[1], Marco D. D. Bicudo[1], and Otto Carlos M. B. Duarte[1]

[1]Grupo de Teleinformática e Automação (GTA)
Universidade Federal do Rio de Janeiro
Rio de Janeiro, RJ, Brazil

[2]Laboratoire d'Informatique de Paris 6 (LIP6)
Université Pierre et Marie Curie - Paris VI
Paris, France

***Abstract*— On most denial-of-service (DoS) attacks, packets with spoofed source addresses are employed in order to disguise the true origin of the attacker. A defense strategy is to trace attack packets back to their actual source in order to make the attacker accountable and isolate him from the network. To date, the proposed traceback systems require either large amounts of storage space on router-connected devices or a sufficient number of received attack packets. In this paper, we propose a new IP traceback system capable of determining the source of every packet received by the victim without storing state in the network infrastructure. For practical purposes, a generalization of the Bloom-filter theory is developed and evaluated. Analytical results are presented to show the efficacy of the proposed system.**

I. Introduction

The current Internet routing infrastructure is vulnerable to anonymous denial-of-service (DoS) attacks [1]. Such attacks are specially designed to conceal the true identity of the attacker in addition to making the services provided by the victim inaccessible to users. These attacks are generally conducted by sending packets to the victim at a higher rate than they can be served, what causes the denial of legitimate service requests. In distributed denial-of-service attacks (DDoS), the aggregate traffic from several different sources is responsible for disabling the services provided by the victim. Recently, the number of distributed attacks against famous websites is alarming and digital plagues have been specifically developed for that purpose [1]. Although less common, denial-of-service attacks constituted by a single packet also exist and are much easier to be conducted [2]. In both cases, the results are financially devastating and a solution that identifies the true origin of attack packets becomes necessary.

Due to the datagram technique employed in the IP protocol, the attacker can inject packets with spoofed source addresses into the network and remain anonymous throughout the attack. In fact, there is no entity or mechanism responsible for verifying the authenticity of the source. Once the routing infrastructure is exclusively based on the destination address, packets with spoofed source addresses generally reach the victim without difficulty. Denial-of-service attacks can also become anonymous due to the stateless nature of IP routing. Currently, no information about forwarded packets is stored in routers for future queries and, as a consequence, it is not possible to deduce the route traversed by a spoofed attack packet.

This work has been supported by CNPq, CAPES, FAPERJ, FINEP, RNP and FUNTTEL.

Several schemes have been proposed for defeating anonymous denial-of-service attacks through IP traceback. The main purpose of IP traceback is to disclose the identity of the attacker by tracing the attack back to its source. Stone [3] proposed an intuitive way for tracing an ongoing attack by observing the interface from which the attack flow comes in every hop. Burch and Cheswick [4] described a flooding technique for detecting from which upstream router the attack packets come. Under another perspective, Savage *et al.* developed a traceback scheme where routers probabilistically insert information about themselves in the packets routed to the victim. After receiving enough attack packets, the victim can reconstitute the entire route. Bellovin [5] suggested a similar approach that employs router-generated ICMP packets instead of inserting information directly into the routed packet. Using high-capacity storage devices connected to routers, Snoeren *et al.* [6] proposed a system capable of tracing a single IP packet by storing digests of every routed packet in Bloom filters [7] located at these devices.

In this paper, we introduce a new approach to the IP traceback problem. We also propose a generalization of Bloom filters [7] and derive its analytical expression. This generalization arises as a solution to evasion techniques that could be implemented if a standard Bloom filter were employed. The proposal consists of using a generalized Bloom filter integrated into the packet for compactly storing the address of each traversed router. Therefore, it is possible to probabilistically trace the complete route traversed by each individual packet. We also show that with the generalized Bloom filter the evasion capability is limited by system parameters. In addition, the traceback process can be started long after the attack is over and without any help from network operators. To date, existing proposals that present equivalent results demand high-capacity storage devices that must be directly connected to routers [6].

The rest of the paper is structured in the following way. In Section II, we introduce the proposed IP traceback system and analyze our proposition of Bloom-filter generalization. Analytical results are then presented in Section III, showing the efficacy of the system. Finally, conclusions and future research work are discussed in Section IV.

II. The Proposed IP Traceback System

This section presents a new IP traceback technique designed to trace the source of each individual packet. The proposal is based on the packet-marking approach to avoid state storage at routers. Instead of using a marking procedure as the one suggested by Savage *et al.* [8], each router inserts a "signature" into the packet, which indicates its presence on the path. A Bloom filter [7] (for a complete understanding of Bloom filters, refer to Appendix I) is employed to reduce the amount of information inserted into the packet and to limit the size of this information to a fixed value to avoid packet fragmentation. In addition, we propose the use of a generalized Bloom filter (Section II-A) to prevent "signature" forgery by the attacker and therefore backtracing failures.

In order to reduce the required space on each packet and to avoid the processing cost of appending data to packets, the attack route is stored in a built-in Bloom filter integrated into the packet. Hence, a static field must then be allocated in the packet header for the Bloom filter. The marking procedure for this case is quite simple. Just before forwarding a packet, the router inserts the IP address of the output interface into the filter. Upon receiving an attack packet, the victim disposes of a filter whose elements are the routers that compose the attack path.

To reconstruct the attack path, the following procedure is used. Initially, the victim checks for the presence of all neighbor routers in the Bloom filter of a received attack packet. The router that is recognized as an element of the filter is identified as the upstream router and is therefore integrated into the attack path. Afterwards, this selected router receives the Bloom filter from the victim and checks which neighbor router is also recognized as an element of the filter, identifying the next upstream router. This process is recursively repeated on each upstream router to reconstruct the actual path traversed by the packet. When a router does not recognize any neighbor router as an element of the filter, the process stops and this router may be considered the source of the attack.

Some advantages come from the adoption of this approach. First, the complete route of each packet can be individually determined. Such behavior is idealized by every IP traceback system since it permits the identification of every source of a distributed attack, even if it contributed with only one packet. By enabling backtracing of a single packet, the system becomes as scalable as it can be. Besides, no information needs to be stored in the network infrastructure. All traceback data is stored at the victim, who chooses to hold it or not according to the local security policy. Another advantage is the ability of tracing an attack long after it is over and without any help from network operators.

On the other hand, additional processing overhead is introduced during each packet routing. Moreover, the adoption of a Bloom filter introduces false positives into the attack path. During the reconstruction procedure, a false positive implies the incorrect integration of a router into the attack path. If this probability is small enough, the occurrence of false positives does not significantly impact on the reconstruction. There would be some concurrent routes for the same packet but the set of possible attackers would still be reduced. Nevertheless, since the attacker controls the initial content of the packet, he can fill all the filter bits with 1. By saturating the filter, every router is integrated into the attack path during the reconstruction procedure, making impractical to distinguish the real path.

In order to minimize misleading techniques and to make the system less dependent of the initial state of the filter, a generalization of the Bloom filter is here proposed. The basic idea of the generalized Bloom filter is to employ both hash functions that set and hash functions that reset bits. We show that with the generalized Bloom filter the false positive probability is reduced and it does not depend so much on the initial condition of the filter. On the other hand, false negatives, which do not exist in standard Bloom filters, are now introduced with this generalization. In the next section, the Bloom-filter generalization is described and an analysis of the false-positive and false-negative probabilities is derived to show the efficacy of this new approach.

A. The Generalized Bloom Filter

As the standard filter, the generalized Bloom filter is also a data structure used to represent a set $S = \{s_1, s_2, \dots, s_n\}$ of n elements in a compact form. It is constituted by an array of m bits and by $k_0 + k_1$ independent hash functions $g_1, g_2, \dots, g_{k_0}, h_1, h_2, \dots, h_{k_1}$ whose outputs are uniformly distributed over the discrete range $\{0, 1, \dots, m-1\}$. The generalized filter is built in a similar way to the standard filter. Nevertheless, the initial value of the bits of the array is not restricted to 0 anymore. In the generalized Bloom filter, these bits can begin with any value. For each element $s_i \in S$, the bits corresponding to the positions $g_1(s_i), g_2(s_i), \dots, g_{k_0}(s_i)$ are set to 0 and the bits corresponding to the positions $h_1(s_i), h_2(s_i), \dots, h_{k_1}(s_i)$ are set to 1. In the case of a collision between a function g_i and a function h_j within the same element, we arbitrate that the bit is always set to 0, $\forall i, j$. The same bit can be set to 0 or 1 several times without restrictions. Figure 1 succinctly illustrates how an element is inserted into a generalized Bloom filter. After inserting the elements, membership queries can be easily made. To check if an element x is in S, we check if the bits of the array corresponding to the positions $g_1(x), g_2(x), \dots, g_{k_0}(x)$ are all set to 0 and if the bits $h_1(x), h_2(x), \dots, h_k(x)$ are all set to 1. If at least one bit is inverted, then $x \notin S$ with high probability. In the generalized Bloom filter, it is possible that an element $x \in S$ may not be recognized as an element of the set, creating a false negative. Such anomaly may happen when at least one of the bits $g_1(x), g_2(x), \dots, g_{k_0}(x)$ is set to 1 or one of the bits $h_1(x), h_2(x), \dots, h_{k_1}(x)$ is set to 0 by another element inserted afterwards. On the other hand, if no bit is inverted, then $x \in S$ also with high probability. This uncertainty is explained by the fact that an element $x \notin S$ may be recognized as an element of the set, creating a false positive. A false positive occurs when the bits $g_1(x), g_2(x), \dots, g_{k_0}(x)$ are all set to 0 and the bits $h_1(x), h_2(x), \dots, h_{k_1}(x)$ are all set to 1 due to a subset of elements of S or to the initial condition of the bit array.

The false-positive probability of a generalized Bloom filter is calculated in a similar way to the standard filter. Given that, in the case of a collision, the functions g_i take precedence over functions h_j, the probability q_0 that a specific bit is set to 0 by an element insertion is the probability that at least one of the k_0 hash functions set the bit to 0; equivalently

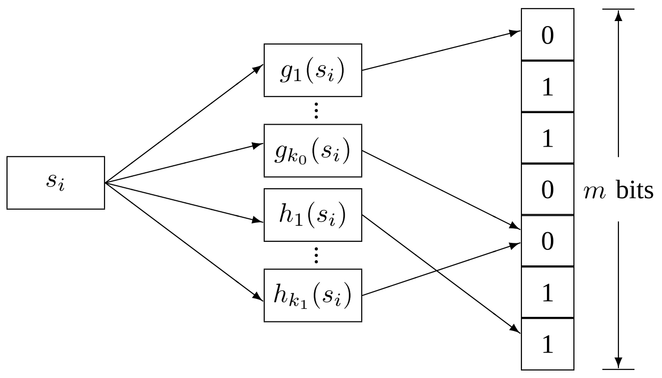


Fig. 1. An element insertion into a generalized Bloom filter.

$q_0 = 1 - (1-1/m)^{k_0} \approx \left(1 - e^{-k_0/m}\right)$. The probability q_1 that a specific bit is set to 1 by an element insertion is the probability that at least one of the k_1 hash functions set the bit to 1 and none of the k_0 hash functions set the bit to 0; thus, $q_1 = \left[1 - (1-1/m)^{k_1}\right](1-1/m)^{k_0} \approx \left(1 - e^{-k_1/m}\right)e^{-k_0/m}$. Finally, the probability that a specific bit remains untouched (not set to 0 nor to 1) during an insertion is just $(1 - q_0 - q_1) = (1-1/m)^{k_0+k_1} \approx e^{-(k_0+k_1)/m}$. Since the same calculation can be made to every bit of the array, on average, a fraction of q_0 bits is set to 0, a fraction of q_1 bits is set to 1, and a fraction of $(1 - q_0 - q_1)$ bits remains untouched on each element insertion. Applying this thought to the bit array, we have on average $b_0 = m.q_0$ bits set to 0, $b_1 = m.q_1$ bits set to 1, and $(m - b_0 - b_1)$ bits untouched on each insertion.

Now, we can determine how the bits are distributed over the array after all insertions. The probability $p_0(n)$ that a specific bit is 0 after n insertions is the probability that the bit is initially 0 and remains untouched by the n elements, or it is set to 0 by the $(n-i)$-th element and remains untouched by the next i elements, $0 \le i \le n-1$. Noticing that $p_0(0)$ represents the probability that a specific bit is initially set to 0, we have

$$\begin{aligned} p_0(n) &= p_0(0)\left(1 - q_0 - q_1\right)^n + \sum_{i=0}^{n-1} q_0 \left(1 - q_0 - q_1\right)^i \\ &= p_0(0)e^{-\frac{(k_0+k_1)n}{m}} + \sum_{i=0}^{n-1}\left(1 - e^{-\frac{k_0}{m}}\right)e^{-\frac{(k_0+k_1)i}{m}} \\ &= p_0(0)e^{-\frac{(k_0+k_1)n}{m}} + \left(1 - e^{-\frac{k_0}{m}}\right)\left(\frac{1 - e^{-\frac{(k_0+k_1)n}{m}}}{1 - e^{-\frac{k_0+k_1}{m}}}\right). \quad (1) \end{aligned}$$

Equivalently, the probability $p_1(n)$ that a specific bit is 1 after n insertions is

$$p_1(n) = p_1(0)e^{-\frac{(k_0+k_1)n}{m}} + \left(1 - e^{-\frac{k_1}{m}}\right)e^{-\frac{k_0}{m}}\left(\frac{1 - e^{-\frac{(k_0+k_1)n}{m}}}{1 - e^{-\frac{k_0+k_1}{m}}}\right) \quad (2)$$

and clearly $p_0(n)+p_1(n) = 1$. On average, a fraction of $p_0(n)$ bits is set to 0 and a fraction of $p_1(n)$ bits is set to 1 after n insertions.

From the bit-array distribution, the probability of a false positive can be easily calculated. Since on average b_0 bits are set to 0 and b_1 bits are set to 1 on each element insertion, the probability of a false positive f_p for the generalized Bloom filter is calculated as follows

$$f_p = p_0(n)^{b_0} p_1(n)^{b_1}. \quad (3)$$

As expected, Equation 3 is reduced to the false-positive probability of the standard Bloom filter when its parameters are used, that is, $k_0 = 0$, $p_0(0) = 1$, and $p_1(0) = 0$. In this case, we have $p_0(n) = e^{-k_1 n/m}$, $p_1(n) = 1 - e^{-k_1 n/m}$, $b_0 = 0$ and $b_1 = m(1 - e^{-k_1/m})$. Noticing that $m \gg k_1$ and using an expansion expression for b_1, we get to the simplification made for the standard filter

$$\begin{aligned} b_1 &= m(1 - e^{-k_1/m}) \\ &= m\left[1 - \left(1 - \frac{k_1}{m} + \frac{k_1^2}{2m^2} - \cdots\right)\right] \approx k_1. \quad (4) \end{aligned}$$

Thus, the probability of a false positive f_p is simplified to the probability of the standard Bloom filter in Equation 13, that is, $f_p = \left(1 - e^{-k_1 n/m}\right)^{k_1}$.

The false-negative probability can be calculated if we have the probability that an specific bit from the $(n-i)$-th element is not inverted by the next i elements, $0 \le i \le n-1$. Thus, the probability $p_{00}(i)$ that a bit set to 0 by the $(n-i)$-th element remains in 0 by the end of the following i insertions is

$$\begin{aligned} p_{00}(i) &= e^{-\frac{(k_0+k_1)i}{m}} + \sum_{j=0}^{i-1}\left(1 - e^{-\frac{k_0}{m}}\right)e^{-\frac{(k_0+k_1)j}{m}} \\ &= e^{-\frac{(k_0+k_1)i}{m}} + \left(1 - e^{-\frac{k_0}{m}}\right)\left(\frac{1 - e^{-\frac{(k_0+k_1)i}{m}}}{1 - e^{-\frac{k_0+k_1}{m}}}\right). \quad (5) \end{aligned}$$

Equivalently, the probability $p_{11}(i)$ of a bit set to 1 by the $(n-i)$-th element remains in 1 by the end of the following i insertions is

$$p_{11}(i) = e^{-\frac{(k_0+k_1)i}{m}} + \left(1 - e^{-\frac{k_1}{m}}\right)e^{-\frac{k_0}{m}}\left(\frac{1 - e^{-\frac{(k_0+k_1)i}{m}}}{1 - e^{-\frac{k_0+k_1}{m}}}\right). \quad (6)$$

We can then calculate the false-negative probability f_n for the $(n-i)$-th element by taking the complement of the probability that none of its bits are inverted. Thus,

$$f_n(i) = 1 - p_{00}(i)^{b_0} p_{11}(i)^{b_1}. \quad (7)$$

As expected, Equation 7 is zero for the standard Bloom filter. In this case, $b_0 = 0$ and $p_{11} = 1$, so independently from other parameters the probability of a false negative is zero. For the last inserted element, the n-th element, the false-negative probability is also zero since no other element can invert any of its bits. In this case, $i = 0$ and $p_{00} = p_{11} = 1$; therefore, the probability of a false negative is also zero.

III. Results

In order to show the advantages of employing a generalized Bloom filter to represent the attack path, we present an analytical comparison between the two versions of our scheme in this section. We compare a simple version that uses the standard Bloom filter and the extended version that uses the new concept of generalized Bloom filter. The analysis comprises three different aspects: false positives, false negatives, and interference of the attacker.

A. False Positives

During the path reconstruction procedure, a false positive implies the integration of an incorrect router into the attack path. Thus, the higher the false-positive probability, the greater the number of possible routes from which the packet may have come, which makes harder the attacker identification.

Figure 2 shows the variation of the false-positive probability of a generalized Bloom filter f_p as a function of $p_1(n)$, according to Equation 3. The probability $p_1(n)$ can be seen as the fraction of bits that are marked as 1 after inserting n elements. For the standard Bloom filter (curve $k_0 = 0$), we can notice that the false-positive probability grows as $p_1(n)$ increases, which is in accordance with Equation 13. The other curves represent the generalized Bloom filter. We can observe that for $p_1(n) = 0$ and $p_1(n) = 1$ the false-positive probability equals zero. It occurs because when we are using at least one function of each type it is required at least one bit marked as 0 and one bit marked as 1 to have a false positive. The result for $k_1 = 1$ and varying k_0 is not shown because it is the dual of Figure 2.

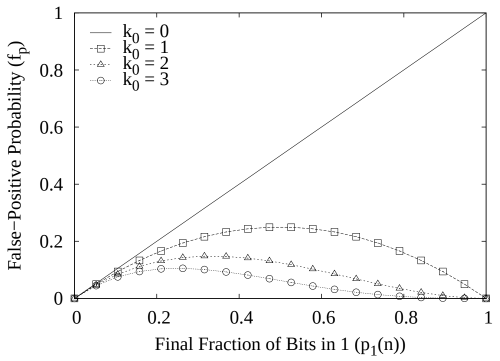


Fig. 2. False-positive probability of a generalized Bloom filter as a function of the final fraction of bits in 1, for $k_1 = 1$.

The maximum false-positive probability of Figure 2 can be calculated by finding out where the derivative of f_p with respect to $p_1(n)$ is zero. Assuming $m \gg k_0$ and $m \gg k_1$, the number of bits marked as 0 and 1 by each element can be expressed by $b_0 \approx k_0$ and $b_1 \approx k_1$, respectively. Therefore, we can simplify Equation 3 by

$$f_p = [1 - p_1(n)]^{k_0}\, p_1(n)^{k_1}. \tag{8}$$

It can be shown that the maximum of Equation 8 occurs for

$$p_1(n) = \frac{k_1}{k_0 + k_1}, \tag{9}$$

assuming $p_1(n) \neq 0$ and $p_1(n) \neq 1$.

When we replace Equation 9 in Equation 8, we find the maximum value for the false-positive probability of a generalized Bloom filter, which is

$$f_p^{max} = \left(\frac{k_0}{k_0 + k_1}\right)^{k_0} \left(\frac{k_1}{k_0 + k_1}\right)^{k_1}. \tag{10}$$

Different from the standard Bloom filter, the generalized version has a bounded false-positive probability f_p^{max}. This value is exclusively determined by k_0 and k_1. This characteristic can restrict the attacker interference in the traceback process, as seen in Section III-C.

B. False Negatives

Using a generalized Bloom filter might lead to false negatives during the attack-path reconstruction procedure. A false negative means not detecting a router by which the attack packet has passed. Therefore, just one false negative is enough to stop the reconstruction procedure and avoid finding the real attack path. It is worth mentioning that the attacker can not interfere in the false-negative probability, as shown by Equations 5, 6, and 7.

The Equation 7 shows the false negative probability for each element $(n - i)$, $0 \leq i \leq n - 1$. In the traceback system, the n-th ($i = 0$) element represents the closest router to the victim and the first element ($i = n - 1$) is the nearest router to the attacker. We can notice that the further the router is from the victim the higher is the false negative probability. Intuitively, we can think that the number of hops between a specific router $(n - i)$ and the victim is the number of routers that can overwrite one of the bits it has set.

Another important observation is that an increase in k_0 or k_1 increases the false-negative probability. It happens because the more functions we use, the higher is the probability of a router to have one of his marked bits inverted by another router. Oppositely, increasing m leads to a lower false-negative probability.

The false negatives are a deficiency of the generalized Bloom filter. For instance, in the case of $k_0 = 1$, $k_1 = 1$, $n = 10$, and $m = 100$, the false-negative probability for the first router ($i = 9$) achieves 15.8%.

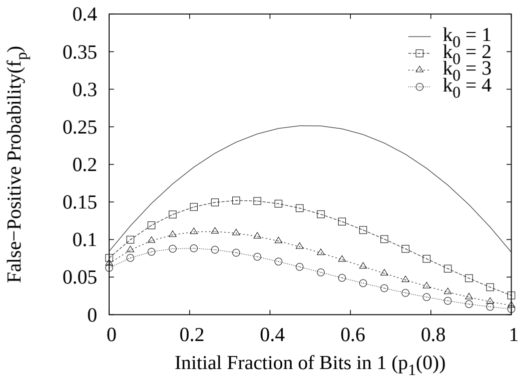
C. Attacker's Interference

Since the filter is integrated into the packet, the attacker may interfere in both standard-filter and generalized-filter systems by setting the initial condition of the filter.

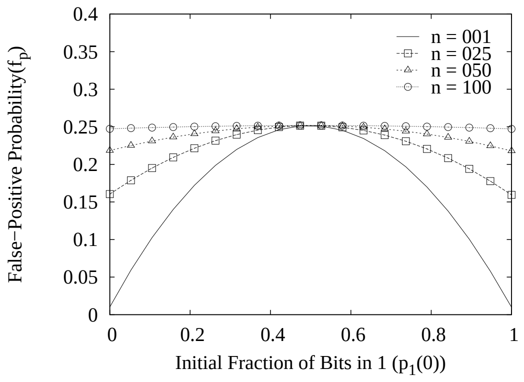
In a standard Bloom filter, the false-positive probability may reach 100% when the attacker just fills with 1 the bits of the packet corresponding to the filter. Nevertheless, when we use a generalized Bloom filter instead of a standard filter, the attacker's interference is considerably reduced. Figure 3 show how the false-positive probability of a generalized Bloom filter is affected by the initial fraction of bits in 1, $p_1(0)$. It can be shown that the value of $p_1(0)$ that maximizes f_p is $k_1/(k_0 + k_1)$, the same value presented in Equation 9. Figure 3(a) corroborates this result. By fixing k_0 and changing k_1, the same result is achieved (not showed). Figure 3(b) shows that as more elements are inserted into the filter, the false-positive probability is less dependent on the initial condition of the filter. We could get to the same result by noticing that the parts of Equations 1 and 2 that depend on the initial condition tend to zero as n increases. In addition, the false-positive probability tends to its maximum value because the fractions of bits in 0 and 1 tend to $k_0/(k_0 + k_1)$ and $k_1/(k_0 + k_1)$, respectively, when n increases, as Equations 11 and 12 show:

$$\lim_{n\to\infty} p_0(n) = \frac{1 - e^{-\frac{k_0}{m}}}{1 - e^{-\frac{k_0+k_1}{m}}} \approx \frac{k_0}{k_0 + k_1}, \tag{11}$$

$$\lim_{n\to\infty} p_1(n) = \frac{\left(1 - e^{-\frac{k_1}{m}}\right) e^{-\frac{k_0}{m}}}{1 - e^{-\frac{k_0+k_1}{m}}} \approx \frac{k_1}{k_0 + k_1}. \tag{12}$$



(a) $n = 10$, $m = 100$, $k_1 = 1$.



(b) $k_0 = k_1 = 1$, $m = 100$.

Fig. 3. False-positive probability of a generalized Bloom filter as a function of the initial fraction of bits in 1.

Therefore, by adopting a generalized Bloom filter instead of a standard Bloom filter, the maximum false-positive probability drops at least 75%, for the worst case where $k_0 = k_1 = 1$.

IV. Conclusion

In this paper, we introduce a new approach to packet-marking IP traceback. Probabilistically, the proposed system is able to trace an attack back to its source analyzing a single packet. Thus, our approach is extremely scalable and fits well to trace each source of a distributed DoS attack. When traversing the network, each packet is marked by routers with a "signature". Thus, when the victim receives a packet, the attack path can be easily identified. A Bloom filter is used to store the router "signatures" in a compact and fixed-size form. Nevertheless, the performance of Bloom filters is very dependable on their initial condition, which is in control of the attacker. Therefore, a generalization of Bloom filters is proposed and employed in the system. We show that with the generalized Bloom filter the misleading ability of the attacker in the traceback procedure is drastically reduced. While for the standard bloom filter the attacker can cause a false-positive probability of 100%, for the generalized Bloom filter the maximum false-positive probability is 25% and this probability is exclusively controlled by design parameters. The tradeoff cost is the introduction of false negatives in the system.

References

[1] *CERT Advisory CA-2003-20 W32/Blaster worm*, Aug. 2003.

[2] *CERT Advisory CA-1997-28 IP Denial-of-Service Attacks*, Dec. 1997.

[3] R. Stone, "CenterTrack: An IP Overlay Network for Tracking DoS Floods," in *9th USENIX Security Symposium*, Aug. 2000.

[4] H. Burch and B. Cheswick, "Tracing Anonymous Packets to their Approximate Source," in *USENIX LISA'00*, Dec. 2000.

[5] S. M. Bellovin, M. D. Leech, and T. Taylor, "ICMP Traceback Messages," *Internet Draft: draft-ietf-itrace-04.txt*, Aug. 2003.

[6] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, B. Schwartz, S. T. Kent, and W. T. Strayer, "Single-Packet IP Traceback," *IEEE/ACM Transactions on Networking*, vol. 10, no. 6, pp. 721–734, Dec. 2002.

[7] B. H. Bloom, "Space/Time Trade-offs in Hash Coding with Allowable Errors," *Communications of the ACM*, vol. 7, no. 13, pp. 442–426, July 1970.

[8] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Network Support for IP Traceback," *IEEE/ACM Transactions on Networking*, vol. 9, no. 3, pp. 226–237, June 2001.

[9] M. Mitzenmacher, "Compressed Bloom Filters," *IEEE/ACM Transactions on Networking*, vol. 10, no. 5, pp. 604–612, Oct. 2002.

Appendix I
The Bloom Filter

In this appendix, the Bloom filter is introduced following the theoretical analysis of Mitzenmacher [9].

The Bloom filter [7] is a data structure used to compactly represent a set $S = \{s_1, s_2, \ldots, s_n\}$ of n elements. It is constituted by an array of m bits and by k independent hash functions $h_1, h_2, \ldots, h_k$ whose outputs are uniformly distributed over the discrete range $\{0, 1, \ldots, m-1\}$. The filter is built by the following rules. First, all the bits in the array are set to 0. For each element $s_i \in S$, the bits corresponding to the positions $h_1(s_i), h_2(s_i), \ldots, h_k(s_i)$ are set to 1. The same bit can be set several times without restrictions. After inserting the elements, membership queries can be easily made. To check if an element x is in S, we check whether the bits of the array corresponding to the positions $h_1(x), h_2(x), \ldots, h_k(x)$ are all set to 1. If at least one bit is set to 0, then $x \notin S$ for sure. Otherwise, $x \in S$ with high probability. This uncertainty is explained by the fact that an element $x \notin S$ may be recognized as an element of the set, creating a false positive. Such anomaly occurs when the bits $h_1(x), h_2(x), \ldots, h_k(x)$ are all set due to a subset of elements of S.

Given that only perfectly independent and uniform hash functions are used, the probability that a specific bit remains in 0 after inserting n elements is $(1 - 1/m)^{kn} \approx e^{-kn/m}$. Since the same computation can be made for every bit in the array, on average, a fraction of $e^{-kn/m}$ bits remains in 0 after all insertions [9]. The probability of a false positive f_p is the probability that we find a bit in 1 for each of the k indicated positions, or

$$f_p = \left(1 - e^{-\frac{kn}{m}}\right)^k. \qquad (13)$$

It is worth mentioning that in Equation 13 it is assumed that the mean number of collisions for each element is near zero, which is valid when $m \gg k$. If this condition is not satisfied, we also need to consider the false-positive probabilities of the cases we have $1, 2, \ldots, k-1$ collisions.