An Improved Single Packet Traceback Scheme for IoT Devices

Jia-Ning Luo[1], Ming-Hour Yang[2]

[1] Department of Information and Telecommunications Engineering, Ming Chuan University, Taiwan
[2] Department of Information and Computer Engineering, Chung Yuan Christian University, Taiwan
deer@mail.mcu.edu.tw, mhyang@cycu.edu.tw

Abstract

Hackers in recent years began to invade the IP camera, CCTV cameras, routers and other IoT devices that embedded Linux operating systems, as the power of the botnet. DDoS attacks will directly affect the operation of the victims, which has a huge impact on the operation of the networks. In current Internet, the attackers can easily forge the source IP addresses, so the system administrator cannot discover the true locations of the intruded devices nor the attackers.

In this paper, we proposed a single packet traceback method that allows the source of an attack to be accurately determined with zero router storage load. We use a 32-bit space in the packet header to record attack paths and use the time to live field to decrease the false positive rate of tracebacks. Our protocol does not require additional storage space on routers for recording attack path data.

Keywords: Packet marking scheme, Packet logging scheme, Hybrid IP traceback

1 Introduction

With the rapid development of the Internet of Things (IoT), all kinds of mobile networking equipment and consumer electronics products are widely distributed on the Internet. According to ABI research estimates, by 2020 there will be more than 41 billion IoT equipments. Most IoT devices (including routers and cameras) are shipped with default passwords and disabled security features. Many customers only install the hardware and not change the default settings, leaving the attacker to leave a major vulnerability to attack. This situation will be accelerated by the number of IoT devices enabled. The IoT device itself can be networked 24 hours a day, causing hackers to love the IoT device to launch DDoS attacks.

Two types of denial of service attacks exist: flood-based attacks and software exploit attacks [1]. Flood-based attacks send massive numbers of packets to overload a target's bandwidth or computational or storage capacities, thus rendering the server unable to accept packets from legitimate users. In contrast, software exploit attacks use fewer packets to attack vulnerabilities in a target system that can disable the system and render it unable to provide services. In addition, most routers do not verify the authenticity of the source IP address, and attackers can forge the source IP address to hide their true locations. Therefore, the development of a traceback scheme to determine the true IP address of attacks is crucial.

Traceback schemes can be classified based on the type of attack they can respond to. Some can only trace the true source of flood-based attacks [2-18], but others can trace the true source of both flood-based and software exploit attacks [19-23], [24-31]. The messaging approach uses additional packets to transmit router information [16-18]. Routers or paths traversed by the packet are marked in additional packets. This can be accomplished by using ICMP messages [16-17] or packets sent to the same destination from different sources [18]. However, the messaging approach requires gathering a large number of packets and is thus not suitable for tracing the source of software exploit attacks. Unlike the messaging approach, the packet marking approach does not require an additional record of the relationship between a tracked packet and a marking packet. However, packet marking approaches can only be used to trace flood-based attacks. In both probabilistic packet marking [2-9] and deterministic packet marking [10-15, 31] schemes, packet marking involves recording a packet's trail by storing router or path identifiers in rarely used header fields. However, these methods require at least eight packets to reconstruct the attack path [12]. Therefore, they are unsuitable for tracing the source of software exploit attacks, in which services can be disabled by a single packet.

Packet logging traceback schemes [19-21] were developed to trace the source of software exploit attacks. In these schemes, each router traversed by a packet records unaltered information from the packet. This resolves the issue of not being able to trace the source from a single packet because of insufficient packet data space. A router's internal logs can be consulted to determine whether a specific packet

*Corresponding Author: Ming-Hour Yang; E-mail: mhyang@cycu.edu.tw
DOI: 10.3966/160792642019052003021

passed through a specific router. For example, Snoeren et al. [19] developed a source path isolation engine that uses Bloom filters to store digests of packets that pass through a router. However, if too many packet digests are stored on a Bloom filter, two separate packets can correspond to the same field and cause false positives. Another problem with packet logging schemes is that they require too much storage space on the router to store packet information. To address this issue, hybrid IP traceback schemes were developed [22-30]. In this type of scheme, unused header fields are used to record a packet's path information. After all header fields have been filled, path information is stored on the router. For example, Yang et al. [27] developed the RIHT scheme in which path information is recorded by encoding the interface numbers of the routers traversed by a packet. When the packet's storage space is exceeded, additional data are stored on the router. The amount of router storage space required is less than 320 kB regardless of the number of packets that pass through a router. Yang [29] also developed the HAHIT scheme in which only the 16-bit identification field in an IP header is used to mark path information. This resolves the issue of not being able to reassemble fragmented packets. In 2014, Yang [30] proposed using multiple tables rather than a single table to reduce the router storage space required by the HAHIT scheme. However, an additional burden is still placed on routers because of the need to store overflow data on the routers.

To allow tracing the source of software exploit attacks from a single packet without incurring any router storage load, Takurou et al. [31] proposed using Bloom filters in the packet header to record information regarding the routers traversed by the packet. Thus, even without requiring additional router storage capacity, a single packet is still sufficient to trace an attacker's true location. However, packet paths that involve too many routers can cause multiple routers to correspond to the same field in the Bloom filter. Zhou et al. [9] modified this approach, using Bloom filters in multiple packets to decrease the chance of field conflicts and thus false positives. However, if the Bloom filter field was overwritten multiple times, the routers cannot be determined in this modified approach, which impacts the accuracy of tracebacks. As a result, this approach cannot be used to determine the source of a software exploit attack from a single packet.

In this paper, we proposed a single packet traceback method that allows the source of an attack to be accurately determined with zero router storage load. We used the Skitter data [32] from the Center for Applied Internet Data Analysis (CAIDA) to perform network topology tests of this traceback scheme. The key contributions of this paper are as follows:

(1) Any traceback can be completed by using only one single packet.

(2) Attacks from multiple sources can be traced simultaneously.

(3) No additional router storage capacity is required.

(4) The traceback scheme has a zero false negative rate.

(5) The traceback scheme has a low false positive rate.

(6) CAIDA's skitter data from 1998 to 2008, comprising network topology constructed from route information obtained by sending packets from a single origin to multiple destinations, were used to validate our method.

In the following sections, Section 2 offers a detailed explanation of the proposed traceback scheme. A more advanced version with fragmentation supported of the proposed scheme is discussed in Section 3. In Section 4, analyses of the scheme are described. Analyses results are discussed in Section 5, as well as a comparison of our scheme and other traceback schemes. Finally, Section 6 concludes the paper.

2 An Improved Single Packet Traceback Scheme for IoT Devices

Our objective was to develop a single packet traceback scheme with zero router storage load capable of attaining zero false negative rates and low false positive rates. To achieve this objective, traceback information is stored only in the IP header. The time to live (TTL) field in the IP header is used to improve traceback accuracy. A detailed description of the proposed scheme is provided later in the paper. First, we must define an attacker model:

- Multiple attackers simultaneously initiate single attacks or multiple attacks from multiple locations.
- Attackers can spoof their IP addresses in the packets (i.e., spoof attacks).
- Attackers are aware of this traceback algorithm and will purposely create a forged mark in the packet to mislead the traceback.

We assume that attackers initiate multiple software exploit attacks or one or more DDoS attacks on a single target. Thus, our scheme can simultaneously trace multiple attack sources. In addition, only the final destination of the packet is considered when routers determine which downstream router to forward a packet to; the packet's originating IP is not authenticated. Attackers can spoof the source IP to hide their location, thus allowing all attacks to reach the victim. Furthermore, because we assume our traceback scheme is openly accessible, attackers can disguise their location by designing a forged mark in packet headers in an attempt to mislead the traceback. To be able to perform tracebacks on attackers with these qualities, and to define the scope of problems that can be resolved by the proposed traceback scheme, our traceback scheme can only work if the following

conditions are met:

- Routers are safe from intrusions.
- Routers can identify whether a packet was forwarded from another router or from a local area network.
- All routers support this traceback scheme.
- The target has an intrusion detection system because an attack must be identified before a traceback can begin.

To simplify assumptions and focus on the main proposal of this paper, we assume that the victim has an intrusion detection system to detect the attacks. To trace the source of an attack, a packet must have space to record the routers traversed by the packet. As shown in Figure 1, we use the 32-bit space in the IP header occupied by the identification, flag, and fragment offset fields to mark and store path information. Because only 0.06% to 0.25% of all packets exceed the maximum transmission unit and must be fragmented [33-34], storing path information in these fields will not affect normal network functioning in most cases.

Offset	0-3	4-7	8-15	16-18	19-31
0	Version	Header length	TOS	Total length	
32	Identification field			Flag	Fragment offset
64	TTL		Protocol	Header checksum	
96	Source IP address				
128	Destination IP address				
160	Options and Paddings (Multiple of 32bits)				

Figure 1. Packet marker field in the IP header

2.1 Packet Marking Scheme

When router R_i receives a packet P, the router first determines whether the packet came from a local area network. If so, router R_i sets packet P's marker field, P.*mark32*, to 0 and packet P's TTL field, *P.ttl*, to the maximum value (255). These values prevent passing a non-zero marker to the Internet, which would result in the inability to accurately determine when to stop tracing and the difficulty of determining whether a packet has exceeded the hot count caused by different initial TTL values. Table 1 lists all symbols used in this method.

In Table 1, we use the IP header's identification field, flag field, and fragment offset field as a 32-bit marking field. The maximum size of a mark is denote as MSM, where MSM can be 2^{15}, 2^{16}, or 2^{32}. The marker field in a packet P is denoted as mark15, mark16, and mark 32, according to MSM. The *h()* is a hash function (for example, MD5) to get the hash value with the input of a router's IP. The bloom filter *BF* computes k distinct packet digests for each packet using independent uniform hash functions and uses the n-bit results to index into a 2n-sized bit array [19].

Table 1. List of symbols

R_i	$\{R_1,R_2,\ldots R_i,\ldots,R_n\}$; a router on the path between the origin and the destination.
P	a packet on the Internet.
P.mark32	a 32-bit marker field in packet P.
P.mark30	a 30-bit marker field in packet P.
P.mark15	a 15-bit marker field in packet P.
P.ttl	the TTL field in packet P's IP header.
IP_{Ri}	the IP address of router R_i.
h()	hash function.
MSM	maximum size of a mark.
BF	the bloom filter.
MS	a flag that indicates the marker field is full and no additional marks can be included in this field.
P.id	an identifier that is identical for all fragments of packet P.
P.no	the sequence of fragments with the same identifier
SF	a flag that indicates the marker field has been fragmented
threshold	the threshold for fragmenting packets
||	the operator 'OR'
%	the operator 'MOD'

As shown in the Algorithm 1, when R_i receives a packet P that is not from a local area network, the router hashes its IP address $\overline{IP_{R_i}}$ to calculate multiple indexes that need to be marked in *P.mark32*. A bitwise OR operation is performed on 1 and the bits referenced by those indexes in *P.mark32*, and packet *P* is forwarded to the next router. This process repeats until the packet *P* reaches its destination.

Algorithm 1. Packet marking algorithm

Input: A Packet header P

1. **if** *P comes from local network* **then**
2. $P.\ mark\ 32 = 0$
3. $P.ttl = 255.$
4. **end**
5. $P.\ mark\ 32[h(IP_{R_i})\%MSM] = 1.$
6. $P.ttl = P.ttl - 1.$
7. Forward this packet to the next router

Figure 2 shows an example in which five senders send packets through eight routers to the same destination. The IP hash values for routers R_1 through R_8 are 5, 1, 3, 3, 7, 1, 5, and 6, respectively.

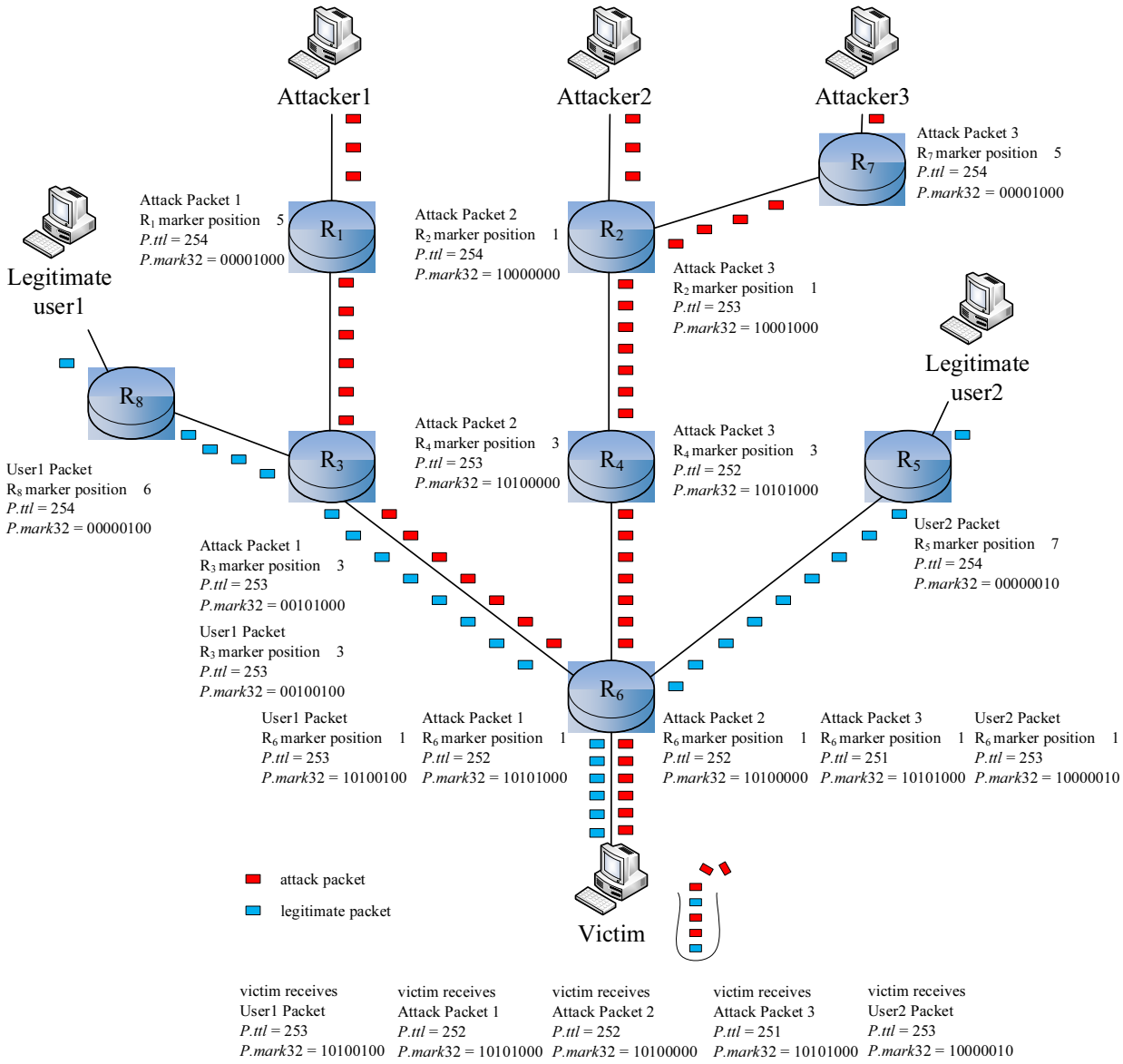

Figure 2. An example of 3 attack and 2 legitimate packets forwarded to the same destination

From Figure 2, we see that attack packet sent by Attacker 1 was passed to the Internet via router R_1 and traversed routers R_3 and R_6 to reach its final destination. Because this packet traversed three routers, its TTL value *P.ttl* is 3 less than the maximum, or 252. The string in *P.mark32*, the packet marker field, is 10101000. Three bits were set to 1 to record that this packet has traversed three routers. Attacker 2's packet also traversed three routers, and its TTL value *P.ttl* is also 3 less than the maximum, or 252. However, because the IP hash values of routers R_2 and R_6 are both 1, only the first and third bits were set to 1, and the string in this packet's *P.mark32* is 10100000. Attacker 3's packet was passed to the Internet via router R_7 but then took the same path as attacker 2's packet to reach the destination. Thus, its TTL value is 1 less than attacker 2's packet, or 251. However, three bits were set to 1 in the packet marker field, and the string in its *P.mark32* is 10101000. During packet marking, the router does not distinguish whether a packet is part of an attack. Thus, the target also receives legitimate packets. For example, in Figure 2, the destination also received packets from Legitimate users 1 and 2 with *P.mark32* strings of 10100100 and 10010000, respectively.

When the target detects a successful DDoS attack from three attacking packets, we use the path reconstruction algorithm explained in the next section to track the attackers.

2.2 Path Reconstruction

When the victim detects an intrusion, the attack packet P is sent to the traceback server to find the source. The traceback server transmits the fields necessary for tracing the source, *P.mark32* and *P.ttl*, to a router R_i that is one hop upstream from the victim for path reconstruction. As shown by the algorithm 2, after R_i receives *P.mark32* and *P.ttl*, it uses its own IP

address, IP_{Ri}, in a hash function to calculate its marker position in a *BF*.

Algorithm 2. Path reconstruction algorithm

Input:

1. **if** $P.\ mark\ 32[h(IP_{R_i})\%MSM]=1$ **then**
2. $P.ttl = P.ttl + 1$
3. Send $=(BF[h(IP_{R_i})\%MSM], P.ttl)$ to traceback router
4. **if** $P.ttl < 255$ **then**
5. Send *P. mark* 32 and *P.ttl* to all upstream routers
6. **end**
7. **else**
8. End traceback
9. **end**

If R_i's marker position is unflagged (i.e., *P.mark32* [h(IP_{Ri})%MSM] = 0), then router R_i is not in the path of the attack packet *P*, and R_i no longer needs to assist with tracing the source of the attack. In contrast, if R_i's marker position is flagged (i.e., *P.mark32* [h(IP_{Ri})%MSM] = 1), then R_i transmits the *BF* that includes R_i's marker position to the traceback server. R_i also checks whether *P.ttl* plus 1 is equal to the initialization value of 255; if so, then R_i is the boundary router for the attacker and the traceback is complete. If not, R_i transmits *P.mark32* and *P.ttl* to all other linked upstream routers to continue tracing the source of the attack. After the traceback server receives all attack packet *BF*s from the routers, the server integrates all *BF* values to find a router combination that completely matches the *P.mark32* in the *BF* and in which the *P.ttl* is equal to 255. This comprises the routing of an attack packet.

Figure 3 shows how this algorithm can be used to reconstruct the path of the three attackers shown in Figure 2. After the victim identifies the three attack packets, it sends a request to the traceback server to identify the source of these attacks. The traceback server transmits the markers of the three attack packets (*P.mark32* = 10101000, *P.ttl* = 253; *P.mark32* = 10100000, *P.ttl* = 253; and *P.mark32* = 10101000, *P.ttl* = 252) to router R_6, which is upstream from the victim (Figure 3).

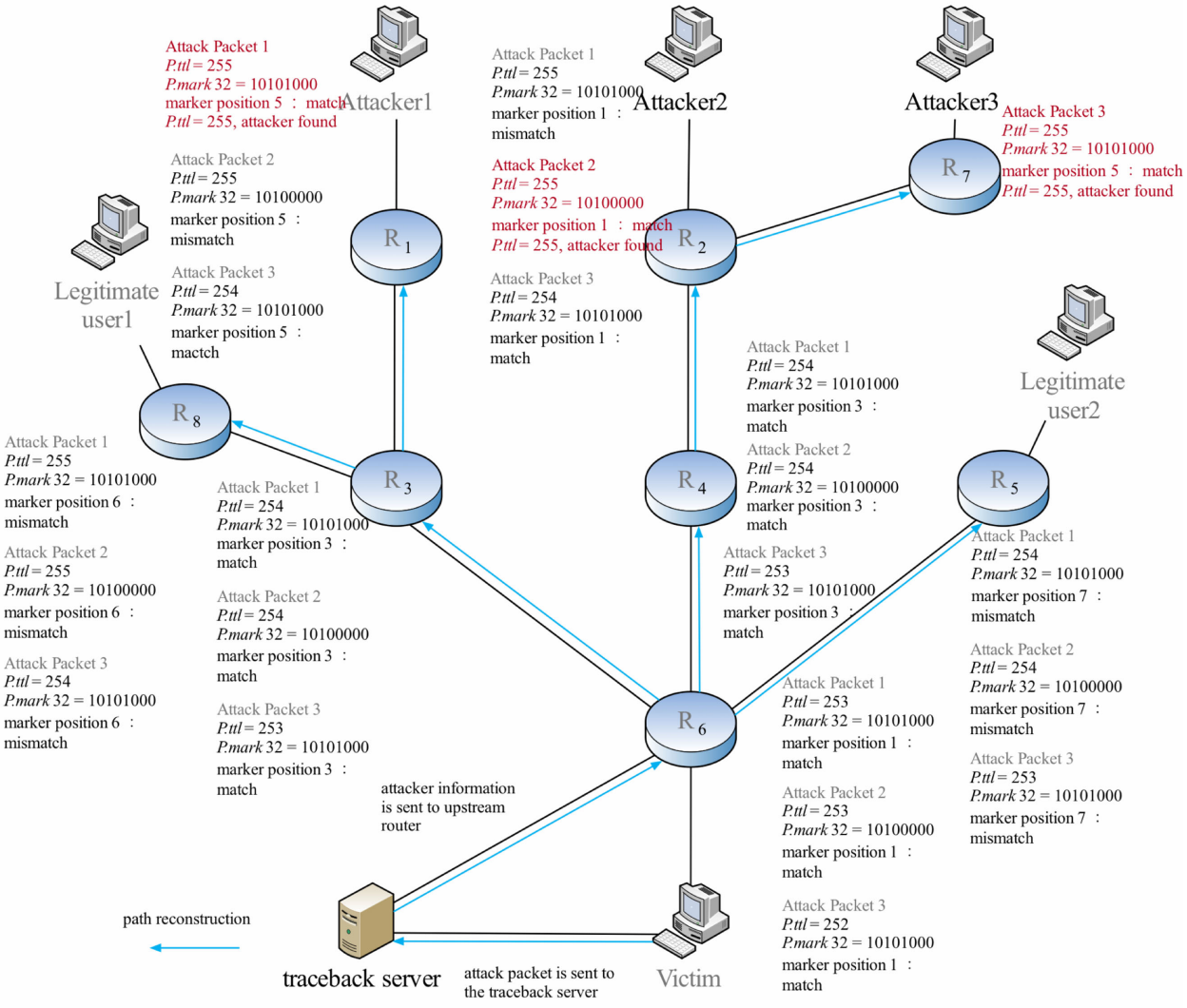

Figure 3. Path reconstruction of the three attacks in Figure 2

Router R_6's marker position is 1 and the first position in all three packets are marked as 1, which indicates that all three packets traversed router R_6. Because the *P.ttl* values of all three packets are lower than 255 at R_6, all three attack packets are sent to the upstream routers R_3, R_4, and R_5 to continue the traceback. R_5's marker position is 7 and the seventh position in all three packets are 0, which indicates that none of the three packets traversed router R_5.

The marking of these three packets match the marker positions of routers R_3 and R_4 and their *P.ttl* values are still lower than 255. Thus, R_3 and R_4 next forward these three packets to their upstream routers R_1, R_8, and R_2 to continue the traceback.

For Attack packet 1, only R_1's marker position matches the marker position in its *P.mark32*. The traceback server combines the *BF*s transmitted by R_6, R_3, and R_1 into *BF* = 10101000. This matches the string contained in attack packet 1's *P.mark32*. In addition, Attack packet 1's *P.ttl* at R_1 is 255. Therefore, attack packet 1 originated from router R_1.

For Attack packet 2, although the traceback server combined the *BF*s transmitted by routers R_3 and R_6 into *BF* = 10100000, the *P.ttl* values transmitted by these two routers were both 254. Thus, the traceback server continues to combine another *BF*, the one transmitted by router R_2, which results in *BF* = 10100000. Thus, the path of Attack packet 2 is R_2 to R_4 to R_6. Without the inclusion of *P.ttl*, the path reconstruction would have stopped at R_3 and R_4. This example shows that *P.ttl* can improve the accuracy of path reconstruction.

For Attack packet 3, the marker positions of routers R_1 and R_2 both match packet 3's marker field; however, packet 3's *P.ttl* is 254. Therefore, the traceback server waits for additional *BF*s to be sent from routers that are upstream to R_1 or R_2. Only R_2 has an upstream router (R_7) that has a matching marker position. Thus, the traceback server successfully reconstructs the path of Attack packet 3: R_7 to R_2 to R_4 to R_6.

3 Traceback Scheme with Bloom Filters that Can Be Fragmented

If a packet's path is too long, most of the marker positions in the marker field may be in use. When this happens, new markings can conflict with existing markings, which could decrease the accuracy of the path reconstruction [19]. Therefore, a method of fragmenting markers is described in this section [9, 18]. By placing marker segments in different packets, the size of the marker field is increased from 32 bits to 240 bits. The single-bit flags MS and SF indicate whether a packet's marker field is saturated and whether it has been fragmented, respectively. As shown in Figure 4, each fragmented packet contains an 11-bit identifier field (id) and a 4-bit sequence field (*no*) to allow reassembly of the packet at the destination. Together, these two fields identify a packet as the n the fragment of the original packet. Finally, a 15-bit field is used to mark the packet marking data.

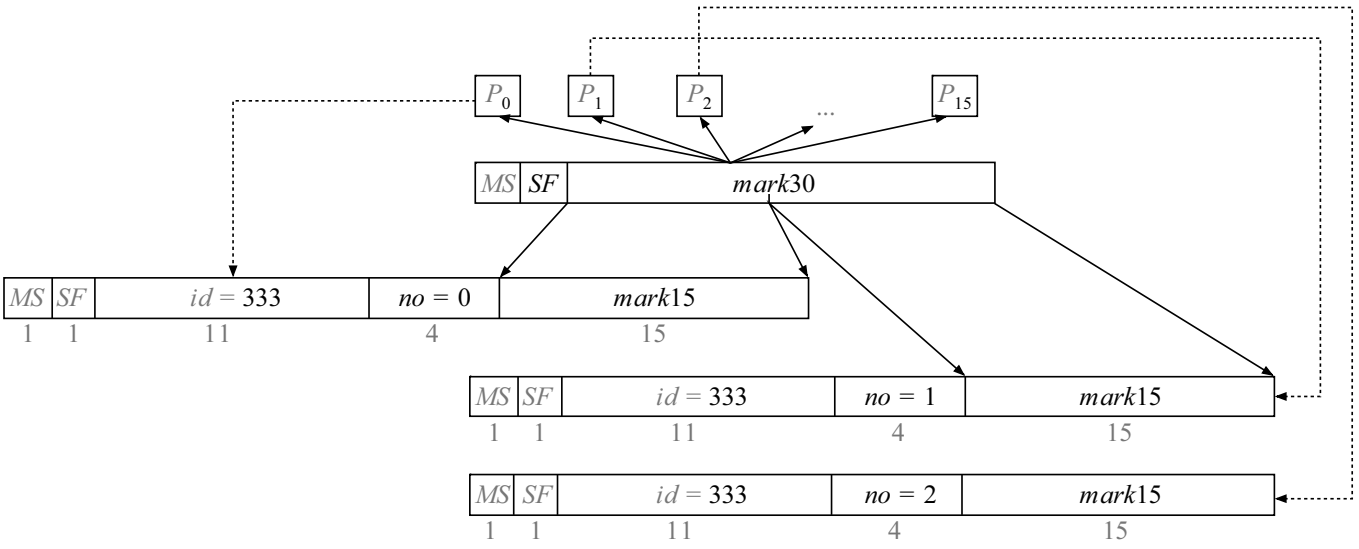

Figure 4. Marker fragmentation of packet P

The marker fragmentation algorithm is shown in Algorithm 3. When a router R_i receives a packet *P*, *P* is handled in one of three ways depending on its marker field.

Algorithm 3. Advanced marking scheme

Input:

1. **if** *P comes from local network* **then**
2. $P.mark30[] = 0$
3. $P.ttl = 255$
4. $P.MkSTRT$ = FALSE
5. $P.SegFlag$ = FALSE

```
6.  end
7.  if P. mark 30 ≤ 2*threshold and
      P.MkSTRT = FALSE and P.SegFlag = FALSE
    then
8.      k = 3
9.      SegP1.id = GetID()
10.     SegP1.no = 0
11.     SegP1.MkSTRT = SegP2.MkSTRT =TRUE
12.     SegP1.ttl = SegP2.ttl = P.ttl
13.     SegP1.SegFlag = TRUE
14.     SegP1.mark15 = P.mark30 [1-15]
15.     SegP2.mark15 = P.mark30 [16-30]
16. elseif |P. mark 15| ≤ threshold and
          P.MkSTRT = FALSE and P.SegFlag =
          TRUE then
17.     k = 2
18.     SegP1.MkSTRT =TRUE
19.     SegP1.id = P.id
20.     SegP1.no = P.no
21  end
22  if P.SegFlag = TRUE then
23      for j ← 2 to k do
24          SegPj.SegFlag = TRUE
25          SegPj.id = SegPj-1.id
26          SegPj.no = SegPj-1.no + 1
27      end
28      SegPj.ttl = 255
29      SegPj.MkSTRT = FALSE
30      SegPj.Mark15 [] = 0
31      SegPj.Mark15 [h(IP_Ri)%MSM] =1
32      SegPj.ttl = SegPj.ttl -1
33  else
34      P.Mark [h(IP_Ri)%MSM] =1
35      P.ttl = P.ttl -1
36  end
37  Forward all packets to the next router
```

In the first scenario, when router R_i receives a packet P, the number of used marker positions in P is less than the threshold value. In this case, R_i's marking is unlikely to conflict with existing markings. R_i uses a hash function of its IP address, IP_{Ri}, to calculate its marker position. A bitwise OR operation is performed on the marker position and 1 and the packet is sent to the next router.

In the second scenario, when router R_i receives a packet P, the number of used marker positions in P exceeds two times the threshold value and P is not flagged as fragmented (i.e., SF = FALSE). As shown in Figure 4, a fragmented marker field requires additional space to store id and no data, which are needed for reassembly but reduces the space available for storing marking information. Therefore, we use the following three steps to create new storage space for marking data. Steps 1 and 2 fragment the original marker field and Step 3 generates a new marker field:

1. First, R_i splits packet *P's mark*30 field in half. The first and second halves are copied to the *mark15* field in packets P_0 and P_1, respectively.

2. Next, R_i generates a unique identifier and writes this to the *id* field in both P_0 and P_1. The values 0 and 1 are written to the *no* field in P_0 and P_1, respectively. R_i also sets both fragmented packets' saturation flags, *MS*, to TRUE to prevent downstream routers from writing marking data into the marker fields. The original packet's TTL value is copied to both P_0 and TTL to assist the traceback server with determining the path length of these two fragmented packets when they are used for source traceback.

3. Finally, a new fragmented packet, P_2, is created. Its *id* field is set to the same identifier generated for P_1 and P_2. Its *no* field is set to 2 and the fragmentation flag *SF* is set to TRUE. R_i*'s* marking data is then written to P_2*'s* marker field. P_2*'s* TTL field is set to the initialization value of and its MS flag is set to FALSE.

In the third scenario, when R_i receives a packet *P,* the number of used marker positions exceeds the threshold value and P is flagged as fragmented (i.e., SF = TRUE). R_i sets the *MS* flag as TRUE, indicating that packet *P,is* saturated and cannot accept additional marking data. As in Step 3 of the second scenario, R_i generates a new fragmented packet P_i with the same identifier as P. Its *no* field is set to *P.no* + 1 and the fragmentation flag *SF* is set to TRUE. R_i *'*s marking data is then written into P_i. P_i's TTL field is set to the initialization value of 255 and its *MS* flag is set to FALSE.

Lastly, one problem remains unsolved: the threshold value for fragmenting marker fields influences the chance of conflicts when marking data is written into the field. This, in turn, impacts the false positive rate for tracing the source of attacks. Section 4 discusses how the threshold value is determined.

When the target wants to reconstruct a path with fragmented markers, all fragmented packets with the same identifier are forwarded to the traceback server. The traceback server begins by forwarding the highest numbered fragmented packet in the sequence (i.e., the fragment with the highest no value) to the routers that are directly upstream from the target. The source router of this fragmented packet is determined using the traceback scheme described in Section 2.2. The previous fragmented packet in the sequence is then sent to the routers upstream from that source router to continue the traceback. Path reconstruction continues until the first two fragmented packets in the sequence (P_0 and P_1) are reached. The marker fields of these two packets must be concatenated. The TTL of P_0 is sent to the routers upstream from the last known router to continue tracing the original source of the attack.

Figure 5 shows an example of marking two distant attack packets with a fragmentation threshold of 3. When Attack packet 1 reaches router R_1, its number of

used marker positions exceeds twice the threshold value and it is not flagged as fragmented (i.e., *SF* = FALSE). Thus, R1 splits the original marker (01111011) in *P.mark*30 in half and copies the strings 0111 and 1011 to the *P.mark*15 fields into the fragmented packets with sequence numbers 0 and 1, respectively. R_i then flags its marker position in a new packet with the sequence number 2, resulting in the string 0001 in *P.mark*15. These three fragmented packets then traverse R_3 and R_6 to reach the final target. The marker fields of P_0 and P_1 remain unchanged, but the marker field of P_2 contains the string 1011.

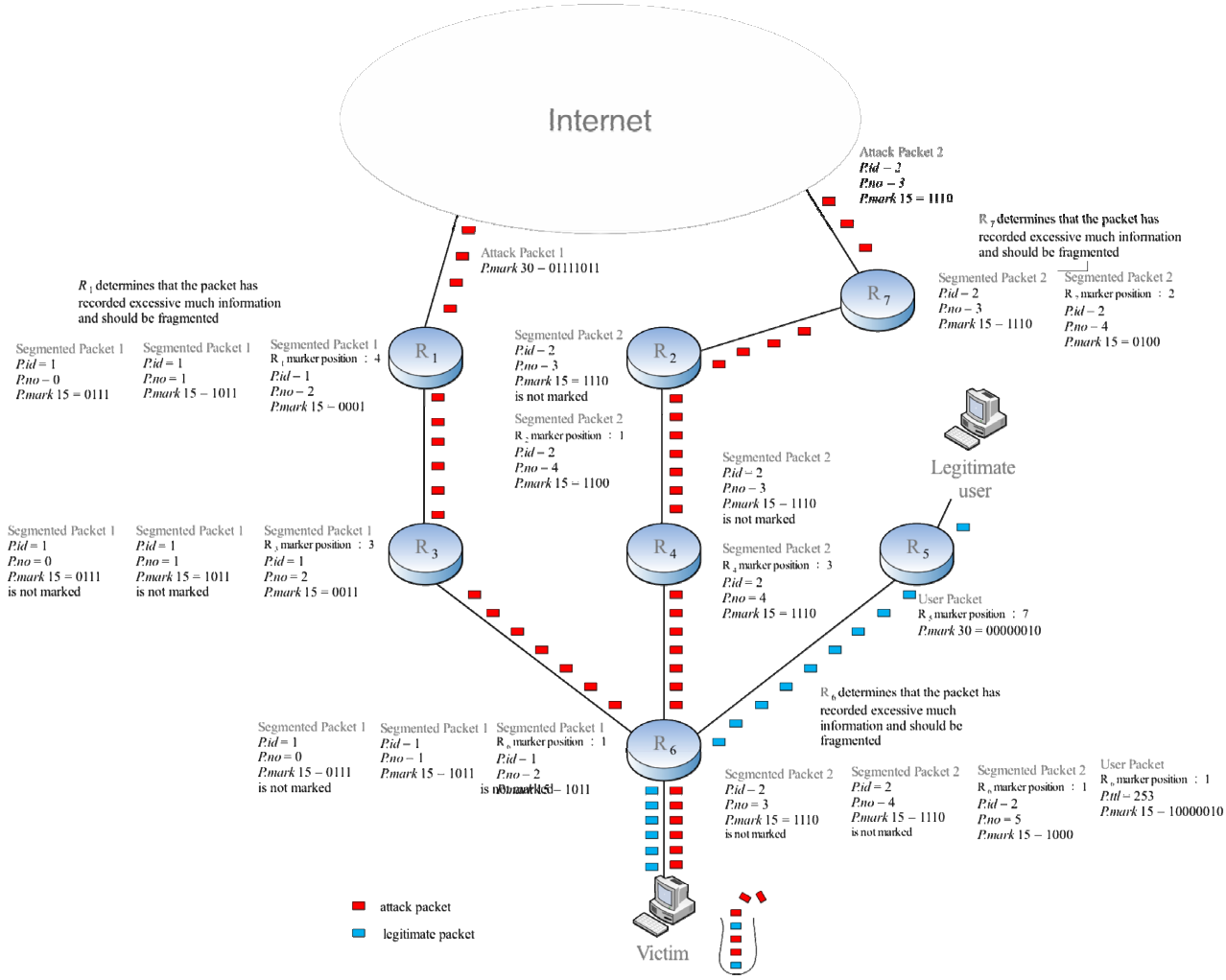

Figure 5. An example of fragmented markers

When Attack packet 2 reaches R_7, its number of used marker positions exceeds the threshold value. The packet flagged as fragmented (i.e., *SF* = TRUE), and its sequence number is 3. Thus, R_7 generates a new fragmented packet with the same identifier as Attack packet 2. The new fragmented packet's sequence number is 3 + 1 = 4, and its *P.mark*15 contains the string 0100. These two packets traverse R_2 and R_4 to reach R_6. At R_6, P_4's *P.mark*15 contains the string 1110.

The number of used marker positions exceeds the threshold value of 3. R_6 then produces yet another new fragmented packet with the same identifier as Attack packet 2 and the sequence number 5 and marks its *P.mark*15 field with 1000. This fragmented packet is sent with all others to the target.Figure **6** contains a path reconstruction example. When the target detects an attack packet, all fragmented packets marked as part of this packet are sent to the traceback server for path reconstruction. The traceback server classifies and sorts these attack packets and starts path reconstruction with the highest numbered fragmented packet. The fragmented packet with identifier 1 and sequence number 2 is sent to R_6 for path reconstruction. R_6's marker position is 1, which matches the string 1011 in *P.mark*15. This fragmented packet is sent to the upstream routers R_3 and R_4, whose marker positions also match the string in *P.mark*15, and the packet is sent further upstream to routers R_1 and R_2 for path reconstruction. When the packet reaches R_1, R_1's marker position matches the string 1011 in *P.mark*15 and the TTL is 255. This indicates that the packet originated from R_1. The traceback server finds that the next packet needed for path reconstruction is the fragmented packet with the sequence number 1. The traceback server concatenates the marker fields from two fragmented packets with identifier 1 and sequence numbers 0 and 1 and sends this to R_1 for path reconstruction.

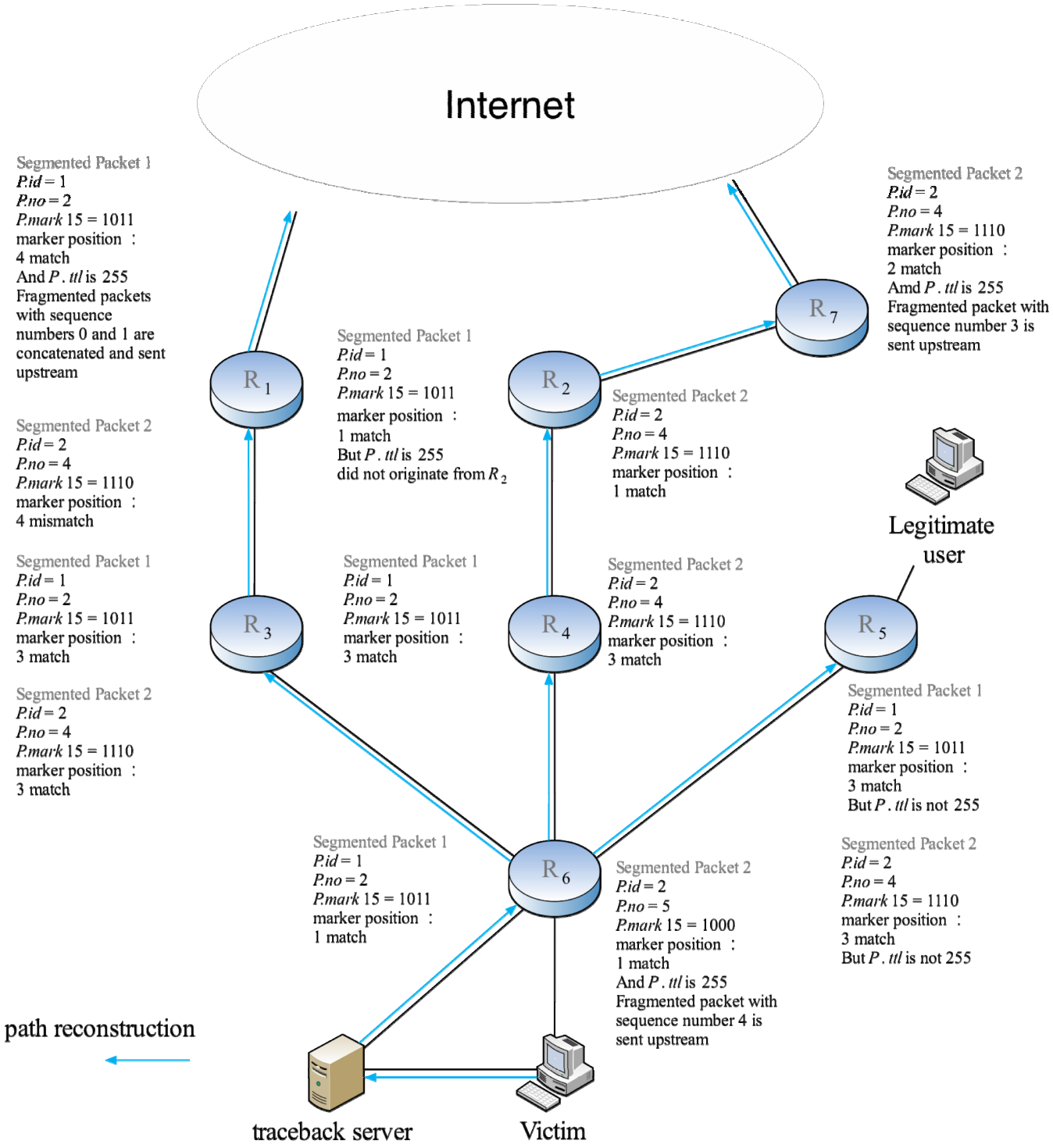

Figure 6. Path reconstruction example

The fragmented packet with identifier 2 and sequence number 5 is sent to R_6 for path reconstruction. R_6's marker position is 1, which matches the string 1000 in *P.mark*15. In addition, the *P.ttl* is 255. Thus, the traceback server forwards the fragmented packet with sequence number 4 to the upstream servers R_3 and R_4. The string in *P.mark*15 is 1110. Both R_3's and R_4's marker positions match the string 1110 in *P.mark*15, and the fragmented packet continues on to the upstream servers R_1 and R_2. When this fragmented packet arrives at R_1, R_1's marker position does not match the string in *P.mark*15. This indicates that this packet did not traverse R_1. R_2's marker position is 1, which matches the string, and the fragmented packet is sent upstream to R_7. R_7's marker position also matches the string, and the *P.ttl* is 255. The traceback server then forwards the fragmented packet with sequence number 3 to routers upstream from R_7 to continue the path reconstruction.

4 Traceback False Positive Rate Analysis

In this section, we analyzed the false positive rates for the proposed traceback scheme when using a

Bloom filter with m bits as the marker field, as shown in Figure 7. When k hash functions are used to calculate the indexes of the packet markers in the marker field, the attack path is recorded in positions referenced by k indexes. In addition, the packet must traverse n routers to reach its destination. Thus, the false positive rate of deducing the source of the packet can be calculated using Eq. (1).

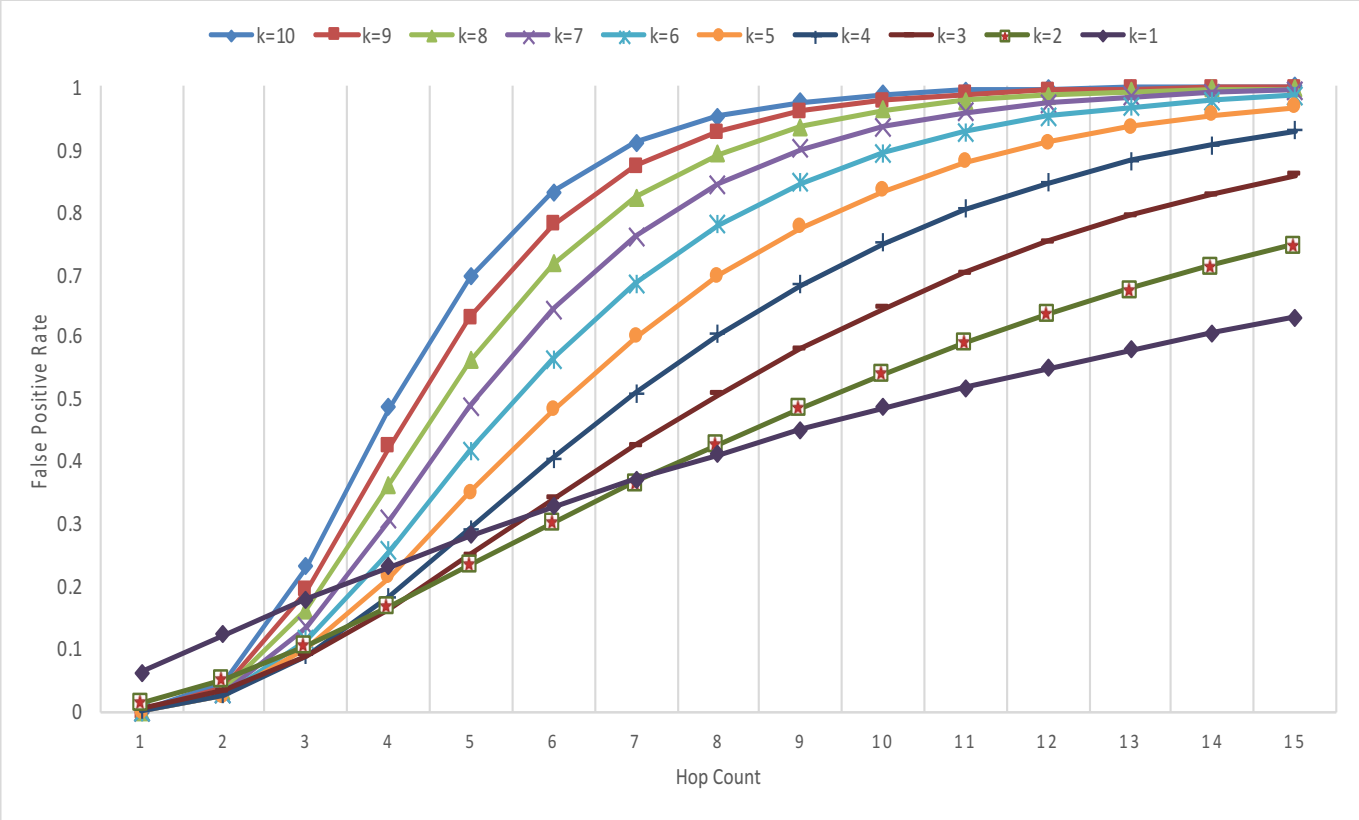

Figure 7. The impact of hop count and hash functions on the false positive rate

$$FP = (1 - [1 - \frac{1}{m}]^{kn})^k \approx (1 - e^{-\frac{kn}{m}})^k \quad \textbf{(1)}$$

We analyzed the skitter data from CAIDA [32], which contains 197,003 complete paths traversing 130,267 routers; the results are shown in Figure 8. We found that the majority of paths had a hop count (n) of 16. Theilmann et al. [35] also found that packets traverse an average of 16 routers (n = 16) to reach their destinations.

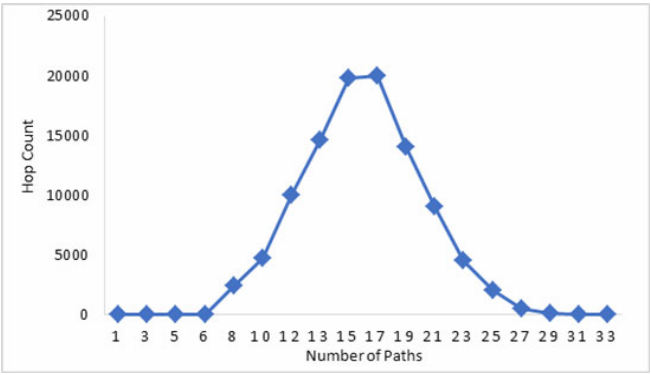

Figure 8. Hop count distribution

We used Eq. 1 to calculate false positive rates for 1 to 10 hash functions (k = 1-10) when the marker field is 32 bits (m = 32) and the hop count is 16 (n = 16); the results are shown in Figure 8. We found that the lowest false positive rate was achieved when k = 1. Thus, we propose that when a small marker field of only 32 bits is used, only one hash function is required to record the attack path.

When marker fields are fragmented because the packets path is excessively long, all downstream routers use a 15-bit marker field (m = 15). To preclude marker position conflicts between two routers, a 15-bit marker field can only be marked by 15 routers before a new marker field is required for the remaining path.

Thus, we used Eq. 1 to calculate the false positive rates for 1 to 15 hops (n = 115) and 1 to 10 hash functions (k = 110); the results are shown in Figure 9. We found that when the fragmentation threshold was low (n < 7), false positive rates were lower when more bits (k) were used for marking. However, this increased the number of packets used. In contrast, when the fragmentation threshold was high (n > 7), using fewer bits (k) for marking resulted in lower false positive rates.

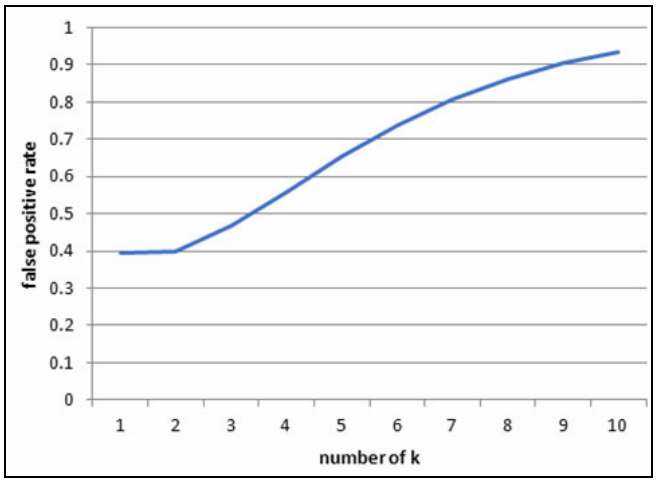

Figure 9. The impact of number of marker bits (k) for each traversed router on the false positive rate

However, one remaining question must be resolved: What is the optimum threshold value for the number of marker positions that can be marked before requiring a new marker field? Because the great majority of packets traverse fewer than 30 routers to reach their destination [35-36] when a router uses C 15-bit marker fields to trace at- tackers, the optimum threshold value is 30/C marker fields. For example, if a 240-bit marking space is used for tracebacks, sixteen 15-bit marker fields are needed to record these markers. Thus, the fragmentation threshold is 30/16 = 2.

5 Experiment

The accuracy of the proposed algorithm is examined in this section. In the following experiment, we used CAIDAs skitter data [32] to generate the network topography and used only one hash function (k = 1) in the bloom filter to mark the packets. The accuracy and efficiency of the proposed scheme were also compared to those of the scheme by Takurou et al. [31] and Long et al. [18].

5.1 Path Reconstruction

Our traceback scheme, like the scheme developed by Takurou et al. [31], records every traversed router in the marker field. Thus, both schemes can achieve the goal of a zero false negative rate. We randomly chose 1000 paths from the 197,003 paths included in CAIDAs skitter data as attack paths. Our 32-bit unfragmented marking scheme and the scheme developed by Takurou et al. were each repeated 10 times to compare the false positive rates. The results are shown in Figure 10. As shown, the false positive rate of our proposed scheme was approximately 92%. It was slightly lower than the false positive rate of the scheme developed by Takurou et al., which approached 100%. The scheme developed by Takurou et al. uses a 32-bit marking space, but conflicts that occur when a packets path is excessively long can cause false positives. Although our scheme uses the TTL field to assist with the traceback, improvements are not evident in a marker field that is as small as 32 bits. Thus, we developed fragmentation of the marker field to improve the false positive rate.

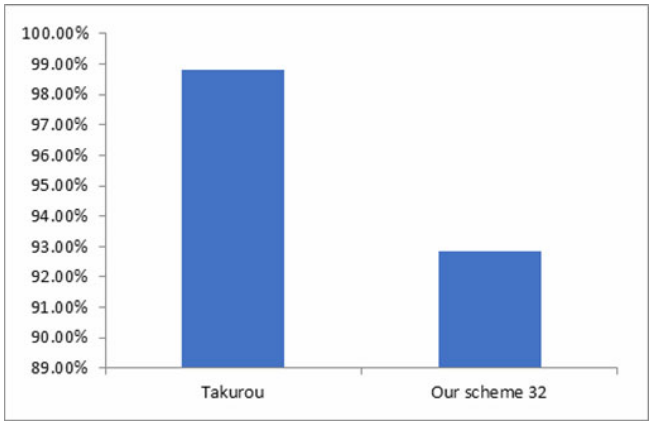

Figure 10. Comparison of the false positive rates from the scheme developed by Takurou et al. and our 32-bit unfragmented marking scheme

We compared the false positive rates of our scheme and the scheme developed by Takurou et al. when using a 120-bit marking space and a 240-bit marking space; the results are shown in Figure 11. Although Takurou et al. did not propose a fragmentation method, we applied one to their scheme to expand the marking space available. Results showed that our scheme, which uses the TTL, greatly improved the false positive rate. A 120-bit marking space requires 8 packets for a traceback; the fragmentation threshold is $\lceil 30/8 \rceil = 4$. The false positive rate was approximately 20%. This was 1/3 of the false positive rate observed in the scheme developed by Takurou et al., which does not use the TTL field. A 240-bit marking space requires 16 packets for a traceback; the fragmentation threshold is $\lceil 30/16 \rceil = 2$. The false positive rate was approximately 6%. This was approximately five times lower than the rate observed in the scheme developed by Takurou et al. Thus, the benefit of using the TTL field increases significantly as the marking space is increased.

Figure 12 shows the relationship between hop count and the false positive rate observed in our scheme. A higher hop count resulted in a higher false positive rate. How- ever, when 120 or more bits were used for the marking space, the increase in the false positive rate as a result of hop count was less noticeable. Thus, our proposed scheme should perform well in current Internet conditions.

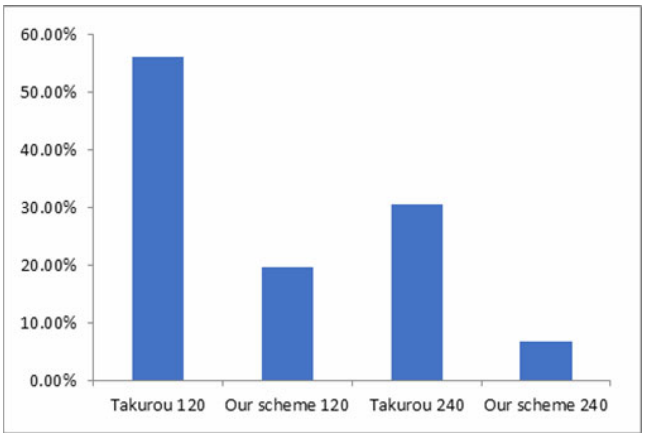

Figure 11. Comparison of the false positive rates

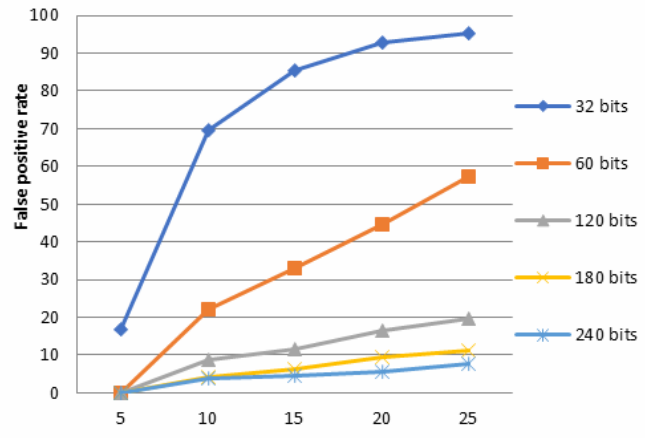

Figure 12. The relationship between hop count and false positive rate

5.2 Marker Delivery Ratio

In this section, the marker delivery ratio observed in our proposed method was compared to those of the opportunistic piggyback marking (OPM) and advanced opportunistic piggyback marking design (AOPM) developed by Long et al. [18]. We randomly chose ten paths with 5 to 25 hops to the same destination from CAIDA's skitter data [32]. One attack packet was sent on each of the paths to the target.

Our scheme fragments the marking data, comprising information from all routers traversed in the path of an attack packet, into 16 fragments. In contrast, OPM and AOPM creates 10 fragments. Figure 13 shows that the marker delivery ratio of our scheme was more than twice that of the scheme developed by Long et al. The schemes developed by Long et al. place markers either in packets with the same destination (the OPM scheme) or in packets with different destinations (the AOPM scheme). Inadequate traffic flow past a particular router creates the problem of insufficient packets for marking data. Skitter data uses the traceroute command at a starting point to construct a tree-shaped network topography that spreads outwards. Leaves are assumed to be attackers and roots are assumed to be targets. These assumptions result in large numbers of packets traversing routers close to the roots and much fewer packets traversing routers close to the leaf. This decreases the marker delivery ratio in the scheme developed by Long et al.

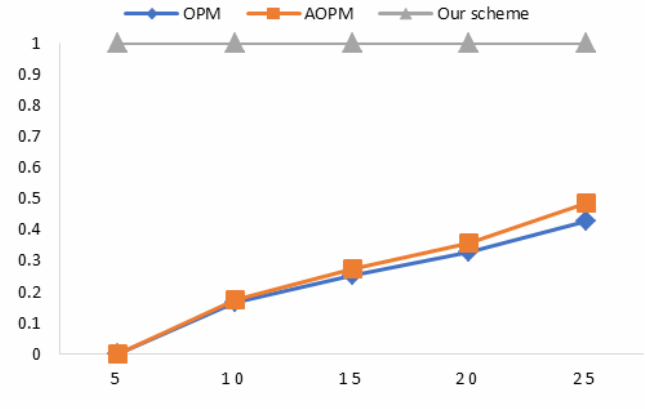

Figure 13. Comparison of marker delivery ratios

Our scheme uses the existing fragmentation ability of routers to produce additional marking space on the same path. Markers are not dependent on network traffic for delivery; thus, our scheme achieved twice the marker delivery ratio that was observed in the scheme developed by Long et al. This is a critical advantage. If intact markers are not delivered to the target when attacks occur, tracing the source of the attacks becomes impossible.

5.3 The Number of Marked Packets Required to Carry the Marking Data

Figure 14 compares the number of marked packets required to record an attack packet in OPM, AOPM, and our scheme. We randomly chose one path with 5 to 25 hops to the same destination from CAIDA's skitter data. An attack packet was sent on this path to the target.

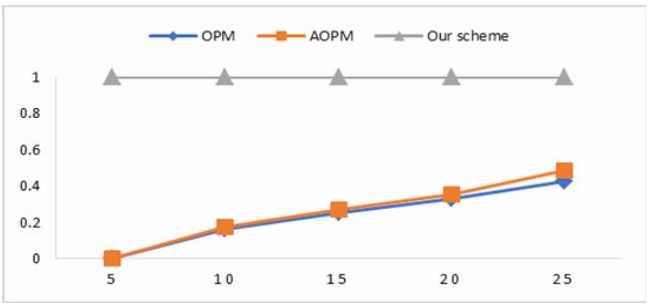

Figure 14. Number of marked packets required

Figure 15 shows that the number of marked packets requires to record an attack packet in our scheme was far fewer than the number required by the two methods. OPM requires 10 marked packets to carry the path information for each router traversed in the path; thus, a total of 250 marked packets were required. AOPM uses packets that have different destinations but share intermediate routers along the path to assist with

sending marker fields to a downstream router. However, the number of marked packets required by AOPM increased slightly to 270. Our scheme does not require assistance of other packets. The number of required marked packets is dependent only on the hop count. If 240 bits of marking space is required and the fragmentation threshold is 2, then a path with a hop count of 25 uses only $\lceil 25/2 \rceil$, or 13 marked packets.

Figure 15. Number of marked packets required to record an attack packet

6 Conclusion

Our proposed scheme, an improved packet traceback scheme with bloom filters, uses a 32-bit space in the packet header to record attack path information. This enables single- packet tracebacks via packet marking and does not require additional storage space on routers for recording attack path data. Because no data is stored on routers, the router load is reduced compared to packet logging or hybrid IP traceback schemes. In addition, using the TTL field in packet headers decreases the false positive rate caused by marker field conflicts. We also proposed a dynamic marking space to further improve upon the traceback accuracy of the 32-bit marker field. Using 120-bit marking space results in a false positive rate of approximately 20%, which is 1/3 lower than the false positive rate observed in the scheme developed by Takurou et al.; using a 240-bit marking space results in a false positive rate of approximately 6%, which is five times lower than that observed in the scheme developed by Takurou et al. Furthermore, our proposed scheme has a 100% marker delivery ratio and only requires 16 packets to trace the source of an attack with 94% accuracy. Our proposed method successfully achieves the objectives of single packet traceback, zero router storage load, zero false negative rate, and low false positive rate.

Acknowledgements

This research was supported by the National Science Council of Taiwan under grant no. MOST 107-2218-E-011-012-, MOST 107-2221-E-033-010-, and MOST 107-2221-E-130-001-.

References

[1] A. Hussain, J. Heidemann, C. Papadopoulos, A Framework for Classifying Denial of Service Attacks, *2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, Karlsruhe, Germany, 2003, pp. 99-110.

[2] A. Yaar, A. Perrig, D. Song, FIT: Fast Internet Traceback, *24th Annual Joint Conference of the IEEE Computer and Communications Societies*, Miami, FL, 2005, pp. 1395-1406.

[3] D.X. Song, A. Perrig, Advanced and Authenticated Marking Schemes for IP Traceback, *2001 Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies*, Las Vagas, NV, 2001, pp. 878-886.

[4] H. Tian, J. Bi, X. Jiang, W. Zhang, W, A Probabilistic Marking Scheme for Fast Traceback, *2010 Second International Conference on Evolving Internet*, Valencia, Spain, 2010, pp. 137-141.

[5] J. Liu, Z.-J. Lee, Y.-C. Chung, Dynamic Probabilistic Packet Marking for Efficient IP Traceback, *Computer Networks*, Vol. 51, No. 3, pp. 866-882, February, 2007.

[6] S. Savage, D. Wetherall, A. Karlin, T. Anderson, Network Support for IP Traceback, *IEEE/ACM Transaction on Networking*, Vol.9, No.3, pp. 226-237, June, 2001.

[7] V. Paruchuri, A. Durresi, S. Chellappan, TTL Based Packet Marking for IP Traceback, *IEEE Global Telecommunications Conference (GLOBECOM 2008)*, New Orleans, LA, 2008, pp. 1-5.

[8] S. Saurabh, A. S. Sairam, Linear and Remainder Packet Marking for Fast IP Traceback, *2012 Fourth International Conference on Communication Systems and Networks (COMSNETS)*, Bangalore, India, 2012, pp. 1-8.

[9] Z. Zhou, B. Qian, X. Tian, D. Xie, Fast Traceback against Large-scale DDoS Attack in High-speed Internet, *International Conference on Computational Intelligence and Software Engineering*, Wuhan, China, 2009, pp. 1-7.

[10] A. Belenky, N. Ansari, Accommodating Fragmentation in Deterministic Packet Marking for IP Traceback, *IEEE Global Telecommunications Conference (GLOBECOM'03)*, San Francisco, CA, 2003, pp. 1374-1378.

[11] A. Belenky, N. Ansari, IP Traceback With Deterministic Packet Marking, *IEEE Communications Letters*, Vol. 7, No. 3, pp. 162-164, April, 2003.

[12] A. Belenky, N. Ansari, Tracing Multiple Attackers with Deterministic Packet Marking (DPM), *2003 IEEE Pacific Rim Conference on Communications Computers and Signal Processing*, Victoria, BC, Canada, 2003, Vol. 1, pp. 49-52.

[13] V. S. Rajam, S. M. Shalinie. A Novel Traceback Algorithm For DDoS Attack with Marking Scheme for Online System, *2012 International Conference on Recent Trends in Information Technology (ICRTIT)*, Chennai, India, 2012, pp. 407-412.

[14] H. Tian, J. Bi, P. Xiao, A Flow-based Traceback Scheme on an AS-level Overlay Network, *2012 32nd International Conference on Distributed Computing Systems Workshops (ICDCSW)*, Macau, China, 2012, pp. 559-564.

[15] V. Aghaei-Foroushani, A. N. Zincir-Heywood, IP Traceback Through (Authenticated) Deterministic Flow Marking: An Empirical Evaluation, *EURASIP Journal on Information Security*, Vol. 2013, No. 5, pp. 1-24, November, 2013.

[16] G. Yao, J. Bi, A. V. Vasilakos, Passive IP Traceback: Disclosing the Locations of IP Spoofers from Path Backscatter, *IEEE Transactions on Information Forensics and Security*, Vol. 10, No. 3, pp. 471-484, December, 2015.

[17] S. M. Bellovin, M. Leech, T. Taylor, ICMP Traceback Messages, *IETF Internet Draft*, pp. 1-18, February, 2003.

[18] L. Cheng, D.-M. Divakaran, W.-Y. Lim, V.-L. Thing, Opportunistic Piggyback Marking for IP Traceback, *IEEE Transactions on Information Forensics and Security*, Vol. 11, No. 2, pp. 273-288, October, 2016.

[19] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, B. Schwartz, S. T. Kent, W. T. Strayer, Single-packet IP Traceback, *IEEE/ACM Transactions on Networking*, Vol. 10, No. 6, pp. 721-734, December, 2002.

[20] L. Zhang, Y. Guan, TOPO: A Topology-aware Single Packet Attack Traceback Scheme, *Securecomm and Workshops*, Baltimore, MD, 2006, pp. 1-10.

[21] E. Hilgenstieler, E. P. Duarte, G. Mansfield-Keeni, N. Shiratori, Extensions to The Source Path Isolation Engine for Precise and Efficient Log-based IP Traceback, *Computers and Security*, Vol. 29, No. 4, pp. 383-392, June, 2010.

[22] C. Gong, K. Sarac, A More Practical Approach For Single-packet IP Traceback Using Packet Logging and Marking, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 19, No. 10, pp. 1310-1324, August, 2008.

[23] K. Choi, H. Dai, A Marking Scheme Using Huffman Codes for IP Traceback, *7th International Symposium on Parallel Architectures, Algorithms and Networks*, Hong Kong, China, 2004, pp. 421-428.

[24] S. Malliga, A. Tamilarasi, A Hybrid Scheme Using Packet marking and Logging for IP Traceback, *International Journal of Internet Protocol Technology*, Vol. 5, No.1-2, pp. 81-91, April, 2005.

[25] S. Malliga, A. Tamilarasi, A Proposal for New Marking Scheme with Its Performance Evaluation for IP Traceback, *WSEAS Transactions on Computer Research*, Vol. 3, No. 4, pp. 259-272, April, 2008.

[26] Y. Wang, S. Su, Y. Yang, J. Ren, A More Efficient Hybrid Approach for Single-packet IP Traceback, *2012 20th Euromicro International Conference on Parallel, Distributed and Network-Based Processing*, Garching, Germany, 2012, pp. 275-282.

[27] M.-H. Yang, M.-C. Yang, RIHT: A Novel Hybrid IP Traceback Scheme, *IEEE Transactions on Information Forensics and Security*, Vol. 7, No. 2, pp. 789-797, April, 2012.

[28] N. Lu, Y. Wang, F. Yang, M. Xu, A Novel Approach for Single-packet IP Traceback Based on Routing Path, *2012 20th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, Garching, Germany, 2012, pp. 253-260.

[29] M.-H. Yang, Hybrid Single-packet IP Traceback with Low Storage and High Accuracy, *The Scientific World Journal*, Vol. 2014, pp. 1-12, February, 2014.

[30] M.-H. Yang, M.-C. Yang, J.-N. Luo, W.-C. Hsu, High Accuracy and Low Storage Hybrid IP Traceback, *2014 International Conference on Computer, Information and Telecommunication Systems (CITS)*, Jeju, South Korea, 2014, pp. 1-5.

[31] H. Takurou, K. Matsuura, H. Hnai, IP Traceback By Packet Marking Method with Bloom Filters, *41st Annual IEEE International Carnahan Conference on Security Technology*, Ottawa, Canana, 2007, pp. 255-263.

[32] CAIDA's Archipelago (Ark) Measurement Infrastructure, https://www.caida.org/projects/ark/, 2014.

[33] I. Stoica, H. Zhang, Providing Guaranteed Services Without Per Flow Management, *Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, New York, NY, 1999, pp. 81-94.

[34] W. John, S. Tafvelin, Analysis of Internet Backbone Traffic and Header Anomalies Observed, *7th ACM SIGCOMM Conference on Internet Measurement*, San Diego, CA, 2007, pp. 111-116.

[35] W. Theilmann, K. Rothermel, Dynamic Distance Maps of the Internet, *Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2000)*, Tel Aviv, Isrel, 2000, pp. 275-284.

[36] R. L. Carter, M. E. Crovella, Server Selection Using Dynamic Path Characterization in Wide-area Networks, *Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution*, Kobe, Japan, 1997, pp. 1014-1021.

Biographies

Jia-Ning Luo holds a Ph.D. degree in Computer Science of National Chiao Tung University, Taiwan. He specializes in network security, operating systems, network administration and network programming. He is currently an associate professor at department of information and telecommunications, Ming Chuan University, Taiwan. His interesting topics includes NFC-based protocols, IoT security and eWallet security.

Ming-Hour Yang received his doctoral degree in Computer Science & Info. Engineering at National Central University, Taiwan. His research mainly focuses on network security and system security with particular interests on security issues

in RFID and NFC security communication protocols. Topics include: mutual authentication protocols; secure ownership transfer protocols; polymorphic worms; tracing mobile attackers.