ABSTRACT

IP TRACEBACK WITH DETERMINISTIC PACKET MARKING (DPM)

by
Andrey Belenky

In this dissertation, a novel approach to Internet Protocol (IP) Traceback - Deterministic Packet Marking (DPM) is presented. The proposed approach is scalable, simple to implement, and introduces no bandwidth and practically no processing overhead on the network equipment. It is capable of tracing thousands of simultaneous attackers during a Distributed Denial of Service (DDoS) attack. Given sufficient deployment on the Internet, DPM is capable of tracing back to the slaves for DDoS attacks which involve reflectors. Most of the processing is done at the victim. The traceback process can be performed post-mortem, which allows for tracing the attacks that may not have been noticed initially or the attacks which would deny service to the victim, so that traceback is impossible in real time. Deterministic Packet Marking does not introduce the errors for the reassembly errors usually associated with other packet marking schemes. More than 99.99% of fragmented traffic will not be affected by DPM. The involvement of the Internet service providers (ISP) is very limited, and changes to the infrastructure and operation required to deploy DPM are minimal. Deterministic Packet Marking performs the traceback without revealing the internal topology of the provider's network, which is a desirable quality of a traceback scheme.

IP TRACEBACK WITH DETERMINISTIC PACKET MARKING (DPM)

by
Andrey Belenky

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy

Department of Electrical and Computer Engineering

August 2003

APPROVAL PAGE

IP TRACEBACK WITH DETERMINISTIC PACKET MARKING (DPM)

Andrey Belenky

Dr. Nirwan Ansari, Dissertation Advisor — Date
Professor of Electrical and Computer Engineering, NJIT

Dr. Krishnan Padmanabhan, Committee Member — Date
Founder, Packet Strategies LLC

Dr. Roberto Rojas-Cessa, Committee Member — Date
Assistant Professor of Electrical and Computer Engineering, NJIT

Dr. Sirin Tekinay, Committee Member — Date
Associate Professor of Electrical and Computer Engineering, NJIT

Dr. Meng Chu Zhou, Committee Member — Date
Professor of Electrical and Computer Engineering, NJIT

BIOGRAPHICAL SKETCH

Author: Andrey Belenky

Degree: Doctor of Philosophy

Date: August 2003

Undergraduate and Graduate Education:

- Doctor of Philosopy in Computer Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2003
- Master of Science in Telecommunication Networks
  Polytechnic University, Brooklyn, NY, 1998
- Bachelor of Science in Computer Engineering
  Polytechnic University, Brooklyn, NY, 1998

Major: Computer Engineering

Presentations and Publications:

Andrey Belenky and Nirwan Ansari,
"Deterministic Packet Marking,"
IEEE/ACM Transactions on Networking, submitted

Andrey Belenky and Nirwan Ansari,
"On IP traceback,"
IEEE Communications Magazine, vol. 41, no. 7, pp. 142–153, July 2003

Andrey Belenky and Nirwan Ansari,
"IP traceback with deterministic packet marking DPM,"
IEEE Communication Letters, vol. 7, no. 4, pp. 162–164, April 2003

Andrey Belenky and Nirwan Ansari,
"Tracing multiple attackers with deterministic packet marking (DPM),"
Proc. of 2003 IEEE Pacific Rim Conf on Communications, Computers and Signal Processing, Victoria B.C. Canada, August 2003, accepted

Andrey Belenky and Nirwan Ansari,
"Accommodating fragmentation with deterministic packet marking (DPM) ,"
Proc. of GLOBECOM 2003, San Francisco, CA, December 2003, accepted

Andrey Belenky and Necdet Uzun,
"Deterministic IP table lookup at wire speed,"
Proc. of INET 99, June 1999

Pinar A. Yilmaz, Andrey Belenky, Necdet Uzun, Nitin Gogate and Mehmet Toy,
"A trie-based algorithm for IP lookup problem,"
Proc. of GLOBECOM 2000, vol. 1, pp. 593–598, November/December 2000

To my wife and my parents

ACKNOWLEDGMENT

I would like to express my deepest appreciation to Dr. Nirwan Ansari, my research supervisor, who guided me through the research providing advice and support on both technical and procedural issues. I would also like to thank Dr. Sirin Tekinay and Dr. Necdet Uzun who used to be my research supervisors throughout my graduate career. Special thanks are given to Dr. Meng Chu Zhou, Dr. Roberto Rojas-Cessa, and Dr. Krishnan Padmanabhan for actively participating in my committee.

I also would like to acknowledge the support from Telcordia Technologies, Jerome Drexler, the New Jersey Commission on Science and Technology, and the New Jersey Commission on Higher Education, that has partially funded my research. Finally, I would like acknowledge the help and support of my peers, namely Yevgeniy Shtutin, Yevgeniy Sakirski, Igor Selivanov, Max Hrabrov, and Marina Sakirski.

TABLE OF CONTENTS

Chapter		Page
1	INTRODUCTION	1
2	PREVIOUS WORKS	3
	2.1 Framework and Evaluation Metrics	3
	2.2 Evaluation of Schemes	9
	2.2.1 Probabilistic Packet Marking (PPM)	10
	2.2.2 ICMP Traceback	13
	2.2.3 Overlay Network	17
	2.2.4 Single-Packet IP Traceback	19
	2.2.5 Controlled Flooding	23
	2.2.6 IP Traceback with IPSec	26
	2.3 IP Traceback State-Of-The-Art Evaluation	29
3	BASIC DPM	31
	3.1 Assumptions	31
	3.2 DPM Principle	32
	3.3 Procedure	33
4	MULTIPLE ATTACKERS AND IP SOURCE ADDRESS INCONSISTENCY	35
	4.1 General Principle of Handling DDoS Attacks	36
	4.2 Single Digest Modification to DPM	36
	4.2.1 Mark Encoding	37
	4.2.2 Reconstruction by the Victim	38
	4.2.3 Analysis	40
	4.3 Multiple Digest DDoS Modification to DPM	44
	4.3.1 Mark Encoding	45
	4.3.2 Reconstruction by the Destination	46

TABLE OF CONTENTS
(Continued)

Chapter **Page**

4.3.3 Analysis 47

5 ACCOMMODATING FRAGMENTATION 52

5.1 IP Fragmentation Background and Terminology 52

5.2 Shortcomings of DPM related to Fragmentation 54

5.2.1 Upstream Fragmentation 54

5.2.2 Downstream Fragmentation 55

5.3 Fragment-Persistent DPM 56

5.3.1 Fundamentals of Handling Upstream Fragmentation with DPM 56

5.3.2 Dealing with Infinite Series 58

5.3.3 Practical Compromise 60

5.4 Size of the *FragTbl* 63

6 TRACEBACK 66

6.1 Types of Cyber Attacks 66

6.1.1 Intrusions 66

6.1.2 Denial of Service Attacks 67

6.2 Traceback Data Structures 69

6.3 Tracing Slaves from Reflectors 70

6.4 Traceback Procedure 72

6.4.1 On Proper *RecTbl*s and Hiding of the Marks 76

6.4.2 On Deleting Marks from *RecTbl* 77

6.5 Conditions for Traceability and Untraceable Attacks 78

6.5.1 Intrusions 79

6.5.2 DoS Attacks 79

6.5.3 Slave-based DDoS Attacks 80

TABLE OF CONTENTS
(Continued)

Chapter **Page**

6.5.4 Reflector-based DDoS Attacks 80

6.5.5 Mixed DDoS Attacks 80

6.5.6 Storage Requirements 82

6.6 Simulation Results 82

6.6.1 Description of the Simulation 83

6.6.2 Description of Profiles 86

7 DPM DEPLOYMENT AND TOPOLOGICAL ISSUES 91

7.1 Structure of the Internet 91

7.1.1 Customer-to-Provider Relationship 92

7.1.2 Peer-to-Peer Relationship 94

7.1.3 Sibling-to-sibling Relationship 95

7.1.4 Backup Relationship 95

7.2 Ideal DPM Deployment 96

7.3 Guidelines for DPM Deployment 97

7.3.1 Illustrative Example 98

8 DISCUSSION AND CONCLUSIONS 104

8.1 Comparison of DPM to Full Path Traceback Schemes 104

8.2 Comparison of DPM to Ingress Address Filtering 108

8.3 IP Traceback Implications and Challenges 110

8.4 Conclusions 111

REFERENCES 112

LIST OF TABLES

Table		Page
2.1	Comparison of Traceback Schemes	30
4.1	Relationship Between Selected k and a, s, d, N_{MAX}, and $E[D]$ for the Single Digest Modification	43
4.2	Relationship Between f, a, k, s, d, N_{MAX}, and $E[D]$ for Selected Combinations for Multiple Digest Modification	50
5.1	Interfaces, Rates and Estimated *FragTbl* Size	65
6.1	Maximum Number of Packets for Marginally Traceable and Untraceable Attacks	82
6.2	Simulation Results for Selected Attack Profiles	90

LIST OF FIGURES

Figure **Page**

2.1 Probabilistic packet marking. 11

2.2 ICMP traceback. 14

2.3 Overlay network. 17

2.4 Single-packet IP traceback. 20

2.5 Controlled flooding. 23

2.6 Using IPSec for traceback. 27

3.1 Deterministic packet marking. 32

3.2 Pseudo code for the basic DPM. 34

4.1 Mark encoding for single digest DDoS modification. 37

4.2 *RecTbl* with k=8, d=10, a=4; mark recording; address recovery. 39

4.3 Pseudo code for the modified single digest DPM algorithm. 44

4.4 Mark encoding for multiple digest DDoS modification. 45

4.5 Address recovery for multiple digest DDoS modification. 46

4.6 Pseudo code for the modified multiple digest DPM algorithm. 51

5.1 IP fragmentation. 53

5.2 Pseudo code for the fragment-persistent DPM. 57

5.3 Pseudo code for the fragment-persistent DPM with fragment counter. . . 59

5.4 Pseudo code for the practial compromise fragment-persistent DPM. 63

6.1 Composition of (D)DoS attacks. 68

6.2 DPM traceback data structure. 69

6.3 Illustration of reflector log requests and responses. 71

6.4 Processing of victim processing of log responses. 73

6.5 Pseudo code for the traceback procedure. 76

7.1 Illustrations of inter AS relationships. 93

LIST OF FIGURES
(Continued)

Figure **Page**

7.2 Ideal DPM deployment. 96

7.3 DPM deployment example, DPM enabled on tier 1 ISPs **A** and **B**. 99

7.4 DPM deployment example, DPM enabled on AS **F**. 99

7.5 DPM deployment example, DPM enabled on AS **H**. 100

7.6 DPM deployment example, DPM enabled on AS **C**. 101

7.7 DPM deployment example, DPM enabled on AS **E**. 101

7.8 DPM deployment example, DPM enabled on AS **D**. 102

7.9 DPM deployment example, DPM enabled on AS **I**. 102

7.10 DPM deployment example, DPM enabled on AS **G**. 103

8.1 Example of partial traceback deployment. 105

8.2 Situation of the subverted router inside an ISP. 106

8.3 Limitations of ingress address filtering. 109

CHAPTER 1

INTRODUCTION

In recent years much interest and consideration has been paid to the topic of securing the Internet infrastructure that continues to become a medium for a broad range of transactions. Currently, Internet security attracts much attention from the industry, academia and even United States (US) congress which held a number of congressional hearings on the subject [1, 2]. A number of approaches to security have been proposed, each attempting to mitigate a specific set of concerns. The specific threat, which is the main focus of this work, is *anonymous attacks*. In anonymous attacks, the identity of the attacker(s) is not immediately available to the victim since the Source Address (SA) field in the attack packets is spoofed. (Distributed) Denial of Service ((D)DoS) attacks are anonymous attacks, which are currently attract the most attention, since there is no obvious way to prevent them or to trace them. The anatomy of DDoS attacks is described by S. Gibson in [3], and his own experiences of being a victim of one, in [4].

Currently, there are several ways of dealing with anonymous attacks. They include source address filtering, SYN Flood Protection, and implementing a BlackHole Router server. Source address filtering, introduced by P. Ferguson [5], prevents packets with the values of the SA field outside preset appropriate range from entering the Internet. If deployed on every ingress interface, this would drastically reduce the number of anonymous packets in the Internet. Unfortunately, source address filtering is associated with high overhead and administrative burden as noted by S. Savage,

et al. [6] and is not widely deployed. SYN Flood protection monitors half-open TCP connection and does not allow more than a certain number of them to exist simultaneously. SYN Flood protection prevents only SYN Flood type (D)DoS attacks and is useless against other types of anonymous attacks. Finally, the ISPs can determine the interface, where the packets of DoS attack entered its network, if the customer reports the attack, by BlackHoling a router on its network as described by UUNET engineers [7]. This method involves human interaction, works only for the backscatter attacks, discussed by D. Moore, G. Voelker, and S. Savage [8], must be performed while the attack is still in progress, and is limited to the boundaries of the given ISP.

The currently available methods for dealing with anonymous attacks are not comprehensive. They either deal with a very limited set of the problems or are too expensive to implement and enforce. While it may be simply impossible to prevent attackers from attempting an attack, it might be possible to lessen, or even completely eliminate, the effects of the attack by not allowing the packets to reach the victim(s). This is the proactive approach discussed in detail by R. Chang [9]. The reality, however, is that prevention of all attacks on the Internet is far from reality. When prevention fails, a mechanism to identify the source(s) of the attack is needed to at least insure accountability for these attacks. This is the motivation for designing IP Traceback schemes.

CHAPTER 2

PREVIOUS WORKS

In this chapter, the current state-of-the-art on IP traceback is presented. The rising threat of cyber attacks, especially DDoS, makes the IP Traceback problem very relevant to today's Internet security. Each approach is evaluated in terms of its pros and cons. Each approach is also related to practical deployment issues on the existing Internet infrastructure. The functionality of each approach is discussed in detail, and then evaluated. The chapter is concluded with a comprehensive comparison of the discussed schemes.

2.1 Framework and Evaluation Metrics

The main objective of this chapter is the evaluation of proposed IP Traceback techniques within a framework originally defined in [10] and presented later in this section. It is worth mentioning that most of the schemes presented here remain theoretical and have not been implemented in the industry, except in trials and simulations. In addition, the following discussion of different traceback methods provides insight for evaluating IP Traceback solutions, which may be proposed in the future.

The following metrics are essential in comparing the IP traceback approaches:

- **ISP Involvement.** Tracking an anonymous attack is not a trivial task. An individual or an organization would find this task difficult, if not impossible, without involving their upstream ISPs. Today, tracing an anonymous attack

even within a single ISP remains a manual task. ISPs and enterprise networks do not have incentives to monitor for attack packets according to R. Chang [9]. The lack of incentives comes from the fact that monitoring for such packets has no immediate benefits to ISP itself and its subscribers. Furthermore, participating in traceback may mean a disclosure of the internal topology, investment in additional equipment, upgrades to the existing equipment, and additional operational cost for the ISP. Consequently, IP Traceback solutions should not assume complete cooperation of ISPs. A desirable quality of an IP Traceback scheme should be *low* ISP involvement, which implies that the scheme should be easily built or inserted with little infrastructure or operational change. Most schemes assume that ISPs will provide limited facility to enable IP traceback, but the burden of the actual traceback process will either be shared between the subscriber and the ISP or will be a sole responsibility of the victim. An Ideal Traceback Scheme would have very low level of ISP involvement

- **Number of Attacking Packets Needed for Traceback.** The attacks can consist of as few as one packet or alternatively can be composed of many thousands of packets. An important evaluation criterion of an IP Traceback scheme is the ability of the scheme to determine the source of the attack based on as few packets as possible once the attack has been identified. This will enable the scheme to successfully traceback more attacks. Ideally, the scheme should be able to trace the attacker with a single packet.

- **Effect of Partial Deployment.** Clearly, if any scheme is adopted on the Internet, not all ISPs will simultaneously implement this function. Clearly, any scheme can perform the traceback only within the ISPs, which deploy it. It is expected that a given ISP would deploy the scheme on all of its routers; however, it is important that the scheme can produce meaningful results when deployed partially within a single ISP. This will allow for partial, gradual deployment on the Internet and will make the scheme more practical. The effects caused by partial deployment can vary from inability to perform tracing altogether to producing meaningful traces limited to the range of deployment, which should be the case for the Ideal Scheme.

- **Processing Overhead.** There are two considerations for processing overhead: *where* it is incurred and *when* it is incurred. Additional processing associated with the traceback scheme can occur on the devices of the ISP network and/or at the subscribers, the potential victims of the attacks. For most methods, additional processing will occur in both places. Processing overhead on the ISP routers is especially undesirable since it may result in the need to upgrade or buy more equipment. Therefore, a scheme with less processing overhead incurred on the network will most likely be accepted by an ISP. For the organizations, additional processing overhead is not so critical. Organizations are usually concerned with the security and are usually willing to invest in dedicated Intrusion Detection System (IDS) servers which would incur most of the processing associated with IP traceback. Another consideration is *when* the

processing overhead is incurred. Processing overhead can be incurred for every packet and during traceback. Preferably it should be incurred only during traceback, which hopefully will be a relatively infrequent operation. An Ideal Scheme would incur minimal processing overhead during traceback only.

- **Bandwidth Overhead.** Additional traffic, which the network has to carry for traceback, is considered as the bandwidth overhead. A large bandwidth overhead is undesirable since it may exhaust the capacity of the links and routers, forcing the ISP to introduce additional capacity and possibly upgrade or purchase new devices. The scheme should not assume availability of infinite bandwidth. As with the processing overhead, the bandwidth overhead can be incurred either constantly, for every packet, or only during the process of traceback, once the attack is identified. It would be preferable to incur bandwidth overhead only during the traceback process, if at all.

- **Memory Requirements.** Additional memory may be required on the routers, or dedicated traceback servers located either at the ISP network or at the client site. Additional memory on the routers is highly undesirable since it may result in upgrades. Additional memory on the dedicated servers is tolerable. Therefore, the important metric of a traceback scheme is the amount of additional memory required on the routers. An Ideal Scheme would have limited amount of additional memory required at the dedicated server, and no additional memory requirements on the network equipment.

- **Ease of Evasion.** The scheme is said to be easy to evade if the attacker, who is aware of the scheme, can easily orchestrate an attack, which will be untraceable. Clearly, this quality is not desirable in a traceback scheme, and the ease of evasion should be as low as possible for an Ideal Scheme.

- **Protection.** Protection refers to the ability of the traceback scheme to produce meaningful traces, if a limited number of network elements involved in a traceback have been subverted. A traceback scheme with good protection would be able to produce valid traces, even if this happens. Taking over a router or a well protected server is an extremely difficult task, and can be accomplished most often due to errors in configuration or improper patching of the software. It is assumed that the devices involved in traceback will be properly managed and protected, minimizing the chances of subversion. High level of protection is preferred in a traceback scheme; however, it is assumed that the probability of the attacker actually taking over a device is very small. An Ideal Scheme should act as if that device was not part of the scheme when a device becomes subverted.

- **Scalability.** Scalability relates to the amount of additional configuration on other devices needed to add a single device to the scheme. It also measures the ability of the scheme to perform as the network size increases. Features which depend on configuration of other devices deteriorate scalability. If only "newly added to the scheme" devices require configuration, the scalability is good. If, on the other hand, introducing another device to the scheme requires configuration

on other devices, the scalability is poor. Also, scalability measures how easy the scheme can expand. An Ideal scheme should be scalable, and configuration of the devices involved should be totally independent of each other.

- **Number of Functions Needed to Implement.** This metric reflects how many different functions a vendor of the equipment needs to implement for a given scheme. It is easier for the vendor to implement fewer functions. Ideally only a single function should be required to be implemented. The amount of effort required to implement each function is not discussed in this dissertation. Most of the functions described are straight forward to implement. It is worth mentioning that historically vendors would implement features on the equipment far ahead of their wide deployment.

- **Ability to Handle Major DDoS Attacks.** This is an extremely important metric which reflects how well the scheme can perform the traceback of DDoS attack under severe circumstances, such as a large number of attackers using reflectors, random address spoofing, etc. Many schemes are not able to cope with all types of attacks. Being able to trace any attack, especially DDoS attacks, is a necessary quality of a traceback scheme. An Ideal Scheme would be able to trace back *all* attacks.

- **Ability to Trace Transformed Packets.** A packet transformation is a modification of the packet during the forwarding process. Most common transformations include Network Address Translation (NAT) specified by P. Srisuresh [11], where Source IP Address and/or Destination IP Address are changed; and

tunneling where a given packet is encapsulated inside another packet. Another type of transformation is packet generation, most common examples of which are Internet Control Message Protocol (ICMP) packets and duplications of the packet in multicast. It is essential for a traceback scheme to handle transformations; otherwise, the attacks, which use packet transformations, cannot be traced. An Ideal Scheme would correctly trace back attacks consisting of packets that undergo any number of transformation of any type.

2.2 Evaluation of Schemes

This section provides an overview on current state-of-the-art approaches to IP traceback, and their evaluate against the metrics established in Section 2.1. The traceback schemes discussed below fall into four general categories as follows:

1. End-host storage
    - Probabilistic Packet Marking (PPM) and variations
    - ICMP Traceback (iTrace)
2. Packet Logging
    - Single-Packet IP Traceback
3. Specialized Routing
    - Overlay Network
    - IP Traceback with IP Security (IPSec)
4. State of the Network Inference
    - Controlled Flooding

2.2.1 Probabilistic Packet Marking (PPM)

This scheme is based on the idea that routers mark packets which pass through them with their addresses or a part of their addresses. Packets for marking are selected at random with some fixed probability of being selected. As the victim gets the marked packets, it can reconstruct the full path, even though the IP address of the attacker is spoofed. Originally proposed by S. Savage, et al. [6], this scheme was improved in several different ways, among which D. Song and A. Perring introduced improved coding methods and protection [12] and D. Dean, M. Franklin and A. Stubblefield came up with algebraic coding methods for PPM [13]. Also T. Doeppner came up independently with a very similar approach [14] at approximately the same time as S. Savage, et al. This scheme is aimed primarily at DoS and DDoS attacks as it needs many attack packets to reconstruct the full path.

Figure 2.1 depicts a schematic illustration of the approach. Attacker **A** initiates an attack to victim **V**. Assume that the path, which the packets take is **R1-R2-R4-R9-R12**. (This path will also be adopted for illustrating other schemes in this chapter.) Each router implementing PPM accepts the stream of packets, and then before routing them, probabilistically marks them with its partial address information (i.e., put the router's partial address in the packet headers). Packets are marked with a marking probability p, which is suggested to be 0.04 in the original proposal [6]. When the victim receives enough of such packets, it can reconstruct the addresses of all the PPM-enabled routers along the attack path. Clearly, in order to reconstruct the full path the flow must contain a large number of packets.

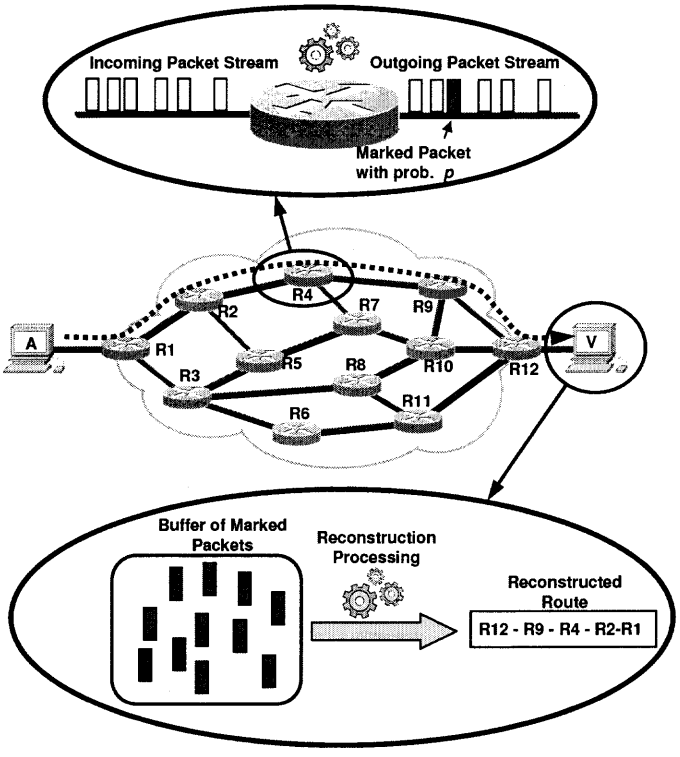

Figure 2.1 Probabilistic packet marking.

To deploy the scheme, the vendors need to implement two functions: marking function and reconstruction function. Once the marking function is available, the software on all routers would have to be upgraded. Upgrade of the software on the routers is a straight-forward task. Once routers are upgraded, PPM would have to be enabled, and that is the extent to which an ISP needs to get involved in the scheme, and therefore the ISP involvement is low. Additional PPM-enabled routers can be added independently, which is an indication of good scalability. The number of packets required for the path reconstruction is measured in thousands for the original proposal by S. Savage, et al. [6], and decreases to just under 1,000 packets for the improved scheme described by D. Song, et al. [12]. For the partial deployment to

be effective, the victim has to be aware of the network topology and routing on the network. Processing overhead in network elements is incurred for every packet. For every packet, the decision is made if it should be marked or not by generating a random number. Additionally, if the packet is marked, more processing overhead will be incurred associated with composing the mark and updating the Identification (ID) field and Reserved Flag (RF) in that packet. The overhead associated with packet marking is minimal, and should not require major upgrades to the router hardware. A major processing overhead will be incurred at the destination during reconstruction. Potentially, the victim would have to perform searches of data structures consisting of billions of entries. Reconstruction data structures will require a large amount of memory as well. However, as it was mentioned in Section 2.1, overhead and additional memory required at the potential victim is not a major setback. Bandwidth overhead for this scheme is zero since all the traceback information is scrambled in the IP packet header and is completely inband. Finally, considering the improved version of PPM described in [12], evasion of the scheme is difficult since marks are authenticated. If a router, which marks the packets becomes subverted, it can be reconfigured to incorrectly mark the packets and still be authenticated by the victim. That may result in an incorrectly reconstructed path. Relying on the assumption made in [12] that the victim is aware of the network topology, subversion of a small number of routers will not be a major problem. Probabilistic packet marking can handle packet modification transformations of the packets directed to the victim. However, in the case of packet generation transformation by a reflector, the traceback will be limited only to that reflector. Fragmented traffic will be corrupted by the scheme,

but the traceback will not be affected. The ID field normally used for fragmentation is used for the mark. If a single fragment of the original datagram is marked, then the reassembly function would fail at the destination. The traceback would still be possible since the mark would be processed before reassembly. This problem is addressed by selecting a lower probability of marking for the fragmented traffic, but this will raise the number of packets needed for the reconstruction. Also, tunneling may create a problem for reconstruction if marks are extracted before the outer header was removed. Carefully choosing the placement for the reconstruction function will remedy this potential problem. All PPM-like schemes are unable to perform the traceback for a major DDoS attacks with a large number of reflectors. The traceback with PPM-like schemes is capable of tracing only a limited number of reflectors.

2.2.2 ICMP Traceback

ICMP Traceback takes a different approach in determining the full path of the attack. This approach was originally introduced by S. Bellovin [15].

Figure 2.2 illustrates the ICMP Traceback scheme. Every router on the network is configured to pick a packet statistically (1 in every 20,000 packets recommended) and generate an ICMP traceback message or *iTrace* directed to the same destination as the selected packet. The iTrace message itself consists of the next and the previous hop information, and timestamp. As many bytes of the traced packet as possible are also copied in the payload of iTrace [15]. The Time To Live (TTL) field is set to 255, and is then used in identifying the actual path of the attack. If routers in the path of the attack from **A** to **V** implement the scheme, then the process

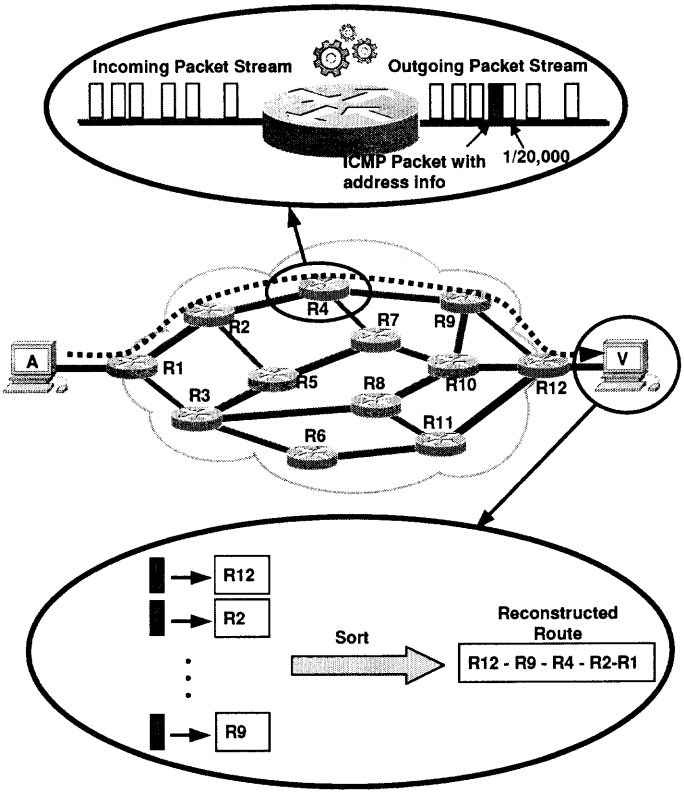

Figure 2.2 ICMP traceback.

illustrated in Figure 2.2 will happen. The routers on the path will generate a new packet with an iTrace message. It is different from PPM where the traceback information was completely in-band. By assuming the victim is under the (D)DoS attack, and therefore the volume of packets going to it is large, the victim will eventually get all the addresses of the routers on the attack path, which implement iTrace. By using TTL fields, these addresses can be sorted to reconstruct the attack path. It was shown by S. Wu [16, 17] that while this approach is efficient and reasonably protected, the chance of receiving useful iTrace is small if the victim undergoes a major DDoS attack, especially so if the attack was carefully orchestrated with the goal of reducing the probability of these useful iTraces. The mechanism to resolve this statistical problem is to associate a weight or value with every iTrace generated. The

value is affected by the distance from the victim, frequency of iTraces being sent to the victim, and the time since the attack has begun. Having these three contributors to the value of iTrace, the original proposal [15] was augmented by the algorithm to make a more educated choice of which packet to select for iTrace. While introducing definite benefits, these augmentations somewhat complicate the algorithm and require a change to the forwarding table on every router implementing this scheme.

The evaluation of this scheme is very similar to the evaluation of PPM discussed in 2.2.1. To deploy the scheme, the vendors need to implement two functions: iTrace function and reconstruction function. It is worth mentioning that implementing value-based ICMP Traceback will require a change to the structure of the routing tables on the routers. Once the iTrace function is available, the software on all routers would have to be upgraded. Upgrade of the software on the routers is a straight-forward task. Once routers are upgraded, ICMP Traceback would have to be enabled, and that is the extent that ISP needs to get involved with this scheme. Additional ICMP Traceback enabled routers can be added completely independently, which is indicative of good scalability. The number of attack packets required for the path reconstruction is measured in thousands since the probability of generating an ICMP traceback message is 1/20,000. For the partial deployment to be effective, the victim has to be aware of the topology and routing on the network. Processing overhead in network elements is incurred for every packet. As in the case of PPM, for every packet the decision is made if it should be marked or not by generating a random number. Additionally, if the packet is marked, more processing overhead will be incurred associated with generating a new packet. The overhead associated

with generating a new packet is minimal and should not require major upgrades to the router hardware. A negligible amount of additional memory is necessary on all the routers if value-based iTraces are implemented. A major processing overhead will be incurred at the destination during reconstruction. Potentially, the victim would have to perform searches of data structures consisting of thousands of entries. Reconstruction data structures will also require a large amount of memory. Bandwidth overhead for this scheme is minimal, and will be about 0.005% derived from the fact that about 1 in every 20,000 packets will be traced. If authentication mechanisms mentioned in [15] are implemented, the evasion of this scheme would be difficult. However, the way the scheme is described, there is nothing which prevents the attacker from generating fake iTraces. The DDoS attacks involve a massive amount of traffic from many different sources; plausible-looking fake chains could easily deceive a victim according to [15]. If a router, which marks the packets becomes subverted, it can be reconfigured to generate incorrect iTraces resulting in an incorrectly reconstructed path. Ability of handling of major DDoS attacks with ICMP Traceback was addressed in [17], where few improvements, described earlier in this section, were suggested to enable the scheme to trace DDoS attacks. However, even with these improvements, ICMP Traceback scheme will not be able to perform the traceback for a DDoS attack with a large number of reflectors. Therefore, the ability to handle major DDoS attacks is poor. Ability to handle packet transformations is very similar to PPM. Transformation undergone by the stream of packets to the victim is not an issue, but transformation caused by a reflector will limit the traceback to the reflector only.

2.2.3 Overlay Network

This solution to the traceback problem is introduced by R. Stone [18]. Logically, the solution introduces a Tracking Router (TR) in the network, as seen in Figure 2.3. This TR monitors all of the traffic which passes through the network. In order to be able to monitor all of the traffic on the network, all packets have to be routed through this TR. This is accomplished by building a Generic Route Encapsulation (GRE) tunnel from every edge router to this TR. Once the appropriate routing has been configured on the edge routers and the TR, all the traffic from an ingress edge router would travel over the GRE tunnel to the TR, and then from the TR, over another GRE tunnel, to the egress edge router. While core routers carry the traffic, logically it is only one hop from an edge router directly to the TR. Shaded network elements in Figure 2.3 are simply transport for the overlay network. This architecture can be visualized as a star topology with the TR in the center, and all of the edge routers on the network connecting to it with GRE tunnels. Since tunnels are built over the existing topology and utilize existing routing protocols, this star-like logical network is said to be an overlay network.

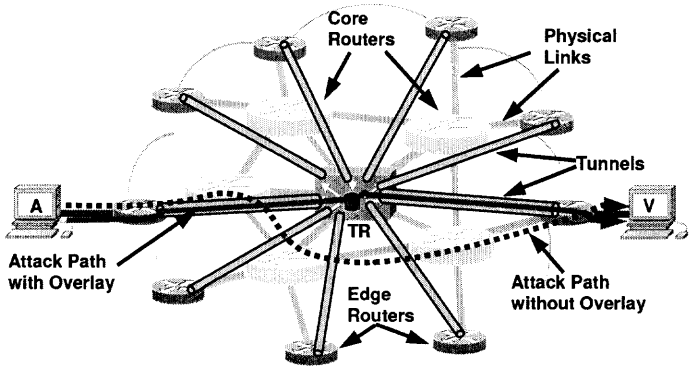

Figure 2.3 Overlay network.

In reality, of course, a single TR will not be able to handle the load of packets from the whole network. Therefore, it is physically a fully connected mesh of several TRs, which can still be logically thought of as a single TR. The TR will utilize signature based intrusion detection. That is different from all the other schemes where the intrusion detection was a function of the victim. When an attack is detected, meaning a single packet, or a sequence of packets, that constitutes an intrusive action, the origin of the attack can be identified because it is only one hop away.

In order to deploy this scheme, no additional functionality needs to be developed by the vendors. The scheme takes advantage of the features available on most of the routers today. On the other hand, ISP involvement for this scheme is large. The ISP has to perform a traceback as well as to identify the attack completely on its own. Also, a number of TRs and IDS servers would have to be purchased by the ISP. Internet service Provider involvement is therefore high. Adding another edge router to the network would result in having to configure the TR in addition to a newly added device to enable the traceback on them. The scheme has a severe limitation. It will only function well within a single administrative domain. In order for the Overlay Network to function well across ISPs, it would be necessary to somehow connect all of the TRs into a single system and that was not proposed by R. Stone [18]. This presents a big scalability issue and constitutes a major limitation to this scheme. A single packet is necessary to traceback any attack, given, of course, that the attack was identified. As soon as IDS on the TR identifies the attack, it can be traced to the end point of the tunnel. Termination of the tunnel will be associated with an interface which faces the network, not the customer. If the edge router has

multiple interfaces facing the customers, it will be impossible to determine from which of those interfaces the attack was initiated. This scheme has a trade-off between the overhead and protection. In the originally proposed configuration with GRE tunnels, the bandwidth is about 20 bytes for each packet. For the attack composed of really short packets, this can be a significant bandwidth overhead. The protection of the scheme is rather low since the tunnel packets can be forged by the attacker when a router is subverted. However, if the tunnels are built with IPSec, the bandwidth overhead is going to be even larger, but the level of protection of the scheme becomes very high. Moreover, some processing overhead on both ends of the tunnel for every packet is incurred. The scheme is able to handle major DDoS attacks in a sense that the source of any packet can be traced to the edge of the network. Handling packet transformations is not an issue for this scheme.

2.2.4 Single-Packet IP Traceback

This approach is introduced by A. Snoeren, et al. [19]. The scheme is officially called Source Path Isolation Engine (SPIE). In hash-based traceback, every router captures partial packet information of every packet, which passes through the router, to be able in the future to determine if that packet passed through it. In this scheme such routers are called Data Generation Agents (DGA). Data Generation Agent's functionality is implemented on the routers. The network is logically divided into regions. In every region SPIE Collection and Reduction Agents (SCARs) connect to all DGAs, and are able to query them for necessary information. The SPIE Traceback

Manager (STM) is a central management unit, which communicates to IDSs of the victims and SCARs, as seen in Figure 2.4.

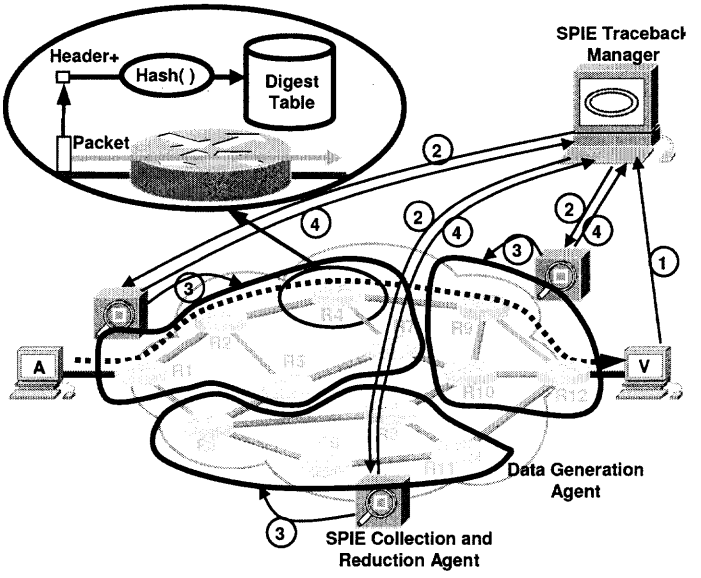

Figure 2.4 Single-packet IP traceback.

As packets traverse the network, the digest of the packets gets stored in the DGAs. In this scheme, constant fields from the IP header and the first eight bytes of the payload of each packet are hashed by several hash functions to produce several *digests*. Digests are stored in a space efficient data structure, called *Bloom Filter*, which reduces storage requirements by several orders of magnitude. When a given Bloom Filter is about 70% full, it is archived for later querying, and another one is used. The duration of using a single Bloom Filter is called a time period. Hash functions also change for different time periods. Also, DGA is able to record any of the transformations such as NAT, IPSec, etc., which may affect those fields. The type of transformation and the data necessary to reconstruct the transformation are stored in the Transform Lookup Table (TLT). Each Bloom Filter for a given time

period has its own TLT associated with it. When the STM receives notification of an attack from the victim's IDS (step 1), it sends the appropriate requests to SCARs (step 2). SCARs in turn obtain copies of the digests and transformation tables from DGAs for the appropriate time period (step 3). After analyzing and correlating the tables, SCARs are able to figure out which routers in the region, if any, forwarded the packet. The SCAR can then reconstruct the path along which the packet traversed through the region, and reports it to the STM (step 4). Based on this information, the STM is able to reconstruct the path through the network.

This scheme involves three functions that have to be implemented: STM, SCAR, and DGA. Internet service provider involvement for this scheme is high. The routers have to be upgraded to support the function of DGA, and the ISP has to purchase equipment for SCARs and at least one STM. The scheme can perform a traceback with a single attack packet. Effects of partial deployment are similar to the case of PPM discussed in Section 2.2.1. If DGA functionality is implemented only on some routers in the ISP network, it is possible to reconstruct the path by checking for the digest at those nodes that implement DGA functionality and extrapolating paths between DGAs that report a hit, provided the topology of the network and routing are known to the STM. Inter-ISP tracing is possible provided there is a necessary degree of cooperation and trust. This issue is briefly mentioned in [19]. Processing overhead is incurred for every packet on every router to store its digests in the Bloom Filter. During traceback, all three functional components incur additional processing. There is no additional processing incurred at the client site. There is no bandwidth overhead associated with every packet; however, there is some minimal bandwidth overhead

incurred during traceback. Additional memory required at DGAs is minimal and is 0.5% of the link bandwidth per unit time, and can be incorporated in the router. A more substantial amount of memory would be required by SCARs and the STM, but these devices are dedicated to the traceback function, and a large amount of memory required for those functions is not a problem. The scheme is extremely difficult to evade. While the scheme is equipped to handle practically any packet transformation, a combination of several packet transformations done in particular order coupled with loss of particular packets may potentially make some transformations irreversible. These conditions are not a major concern because they are unlikely to occur. A subverted DGA can be potentially reconfigured to report that it has seen the packets, which never passed through it, and vice versa. This will produce paths with one hop in error. Subverting a DGA will not, however, allow the attacker to learn of any packet content since nothing can be learned by examining the content of the Bloom Filter. If SCAR becomes subverted, the whole segment of the path can be incorrect. Finally, if STM becomes subverted, the traceback will not produce correct traceback altogether. Reiterating the fact that taking control over a device is extremely difficult, these considerations should not be the major factor. In order to add another DGA, a SCAR needs to be reconfigured, and in order to add another domain to the hierarchy, the STM would have to be reconfigured. Owing to the fact that configuration of other devices is involved when adding another device, the scheme does not scale very well. The scheme is able to handle massive DDoS attacks. The limitation of the scheme is the timing issue. For the high rate interfaces, traceback has to be performed within a very short period of time. The problem is magnified for the inter-domain case

when time synchronization cannot be expected. Also, such strict timing constraint on traceback prohibits "post-mortem" traceback, the traceback long after the attack has finished. This becomes important, when the victim does not realize that it is being attacked, or cannot contact the STM during the attack for some reasons.

A similar but inferior scheme was proposed by S. Matsuda and T. Baba [20, 21].

2.2.5 Controlled Flooding

Controlled Flooding approach to traceback is introduced by H. Burch and B. Cheswick [22]. It is only valid for DoS attacks. It relies on the fact that during DoS attacks the links of the attack path should be heavily loaded. This assumption may not hold for modern backbone networks with abundant bandwidth available on the links. By carefully measuring the incoming traffic to the attacked system and loading the links of the suspected path even more, the drop in the rates of the attack packets should be observed. The process can be repeated for the next hop, and so on until the source of the attack is identified.

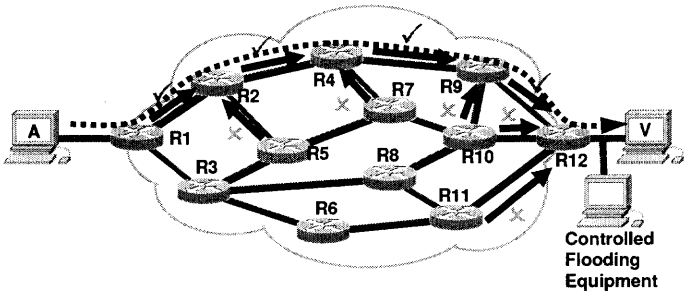

Figure 2.5 Controlled flooding.

This concept is illustrated in Figure 2.5. Once the DoS attack based on flooding is identified by **V**, equipment, which measures the load on the link and equipment, will be used to generate traffic on the network. Once this is accom-

plished, the traceback begins. Routers, which connect to **R12**, the router closest to the victim, are determined. Then, the short burst of traffic is generated from **R11** hoping that the rate of the attacking packets to the victim will drop. In this particular case, it did not. So, this link, and all the paths, which may utilize it are excluded from the set of possible paths. Second, the link from **R10** to **R12** is loaded. Again, no drop in the rate of packets to the victim is observed, and therefore the link is also excluded. When the link between **R9** and **R12** is loaded, the desired drop in the rate is observed. It is thus concluded that the link between **R9** and **R12** belongs to the attack path. The process is recursively repeated until the source of the attack, or the nearest router to the source, is identified.

The way the links are suggested to be loaded is by using the `chargen` service on the routers. The originator of the `chargen` service opens connection to a device on TCP or UDP port 19. In response, this device will generate a large amount of data back to the originator. This outcome is not desirable since the task here is to only load a single link. In order to avoid this, the source address of the equipment is spoofed to be the next hop address with respect to this router. In order to load the link between **R2** and **R4**, Controlled Flooding Equipment would spoof its source address to be the interface of **R4** connecting to **R2**, and start the `chargen` service on **R2**. The packets generated would be directed to **R4**, thus loading the link between these two routers.

There are several limitations to this approach. First of all, contrary to the claim in [22], most routers have `chargen` disabled. In fact, it comes disabled by default now on most of the equipment. Secondly, the approach assumes accessability

to routers on the ISP network. This is also a big assumption. Even if the routers on the ISP network are publicly addressable, it is very unlikely that the customer would be allowed to access them in any way. Such readily available access would be constantly exploited by hackers. This method of denying service is easier. In addition, the authors suggested to basically initiate DoS attacks on the network, although brief ones, in order to determine the source of the similar attack.

On the positive side, however, this is the only method introduced so far that does not rely on any ISP cooperation. This is an important and desirable quality of an ideal traceback scheme. Only a single function must be created to perform Control Flooding. The number of packets required for the scheme to successfully complete a traceback is large. Processing overhead is incurred only during the traceback and only at the equipment of the victim. The bandwidth overhead is extremely high. Additional memory requirements are very limited and will be required only at the victim's site. Partial deployment issue is not applicable here since equipment needs to be deployed only when and where the attack is happening. Ease of evasion and protection of this method are not an issue since there is no threat of compromising the traceback data. The scheme is not sensitive to packet transformations. Only DoS attacks can be traced with this scheme since traceback is limited to one attacking stream. Therefore, this scheme is not able to trace large scale DDoS attacks. Internet service providers, as mentioned above, are not involved with this scheme. While Controlled Flooding can be automated, the authors made a point in [22] that it is preferred to remain manual because of potentially severe consequences of a programming error. For this reason alone, this approach is not feasible for its

wide deployment in addition to the limitations mentioned above. Needless to say that this scheme cannot trace the attack when it is over. It is possible that certain customers may engage in controlled flooding; however, it is absolutely not feasible for ISPs to encourage or support such efforts.

2.2.6 IP Traceback with IPSec

This approach is introduced by a group of scientists headed by H. Chang [23, 24] as part of a network based intrusion detection framework called *DECIDUOUS*. While the framework itself is beyond the scope of this article, the mechanism of identifying the source address of the attack is of interest.

The mechanism is based on an assumption that complete topology of the network is known to the system. What follows is the underlying principle: if there is an IPSec security association between an arbitrary router and the victim, and the attack packets detected are authenticated by the association, then the attack is originated on some device further than this router; if the packets of the attack were not authenticated by this security association, then the attack is originated on some device between this router and the victim. By establishing these security associations, it is possible to identify a single router or a group of routers where the attack was initiated from.

In Figure 2.6, when the attack is detected, an IPSec security association is built between **R4** and **V**. If **A** was in fact an attacker, then attack packets would have to be authenticated since they will go through the tunnel. Next, the tunnel from **R1** to **V** is built. Note that from **R4** to **V** there will be two tunnels encapsulating

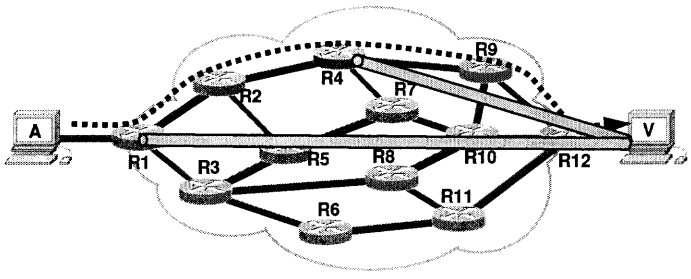

Figure 2.6 Using IPSec for traceback.

traffic from **A**. In reality, (this is not obvious from the figure,) the second tunnel will be encapsulated in the first tunnel. Since the traffic is authenticated by two security associations, it is clear that the attack had originated from somewhere behind **R1**. If, for example, the attack packets were only authenticated by the first tunnel, and not the second, it would mean that the attack comes from somewhere between **R1** and **R4**; in the case of Figure 2.6, it is **R2**.

There is a valid question on how the system determines which routers should the victim build the IPSec associations with, if the source address is not known. The answer is not simple. In short, the system goes through many iterations considering every possible path. Interested readers can familiarize themselves with the intricacies of those algorithms in [23, 24].

No new functionality needs to be developed by the vendors to enable this scheme, since it uses IPSec which is available on most of the routers. An ISP has to get involved in a sense that it must disclose its topology to all of its clients, so they can build IPSec tunnels to all the routers. All the routers on the network have to be configured to be able to build the IPSec tunnels with all the clients. The scalability of the scheme is therefore low. If the "shared secret" type of authentication

is used, all end systems would have to be notified of any change on any router in the network resulting in unacceptable scalability. It is assumed that Digital Certificates will be used for authentication of the Security Association. With the latter method of authentication, the scalability is improved, but still not acceptable for wide inter ISP-deployment. The number of packets necessary for traceback is low. The victim has to build several IPSec tunnels and receive at least one packet after a given tunnel is built in order to traceback the attack. While not discussed explicitly, there is no reason why the scheme cannot be deployed partially only on some routers. Deploying it on some routers would require, however, the knowledge of those routers to all potential victims in advance. The processing overhead is high due to the processing associated with setting up the tunnels with digital certificates in real time, both at the victim's site and on the routers. This overhead will only be incurred during traceback. Bandwidth overhead is potentially high, so is that of all schemes which involve IPSec. This scheme is very difficult to evade because IPSec is very secure. Protection of this scheme is high since the worst thing that can happen if a router becomes subverted is that the IPSec tunnel between this router and the client would be impossible to build. This is equivalent to the situation when this router was not involved in the scheme to begin with. This scheme is not sensitive to most practical packet transformations. Also, the scheme is capable of tracing major DDoS attacks by tracing paths one by one; however, there is an issue with DDoS attacks. The routers can become the target of the attack themselves. Recall that the routers in the ISP have to be open for clients to setup IPSec tunnels. This can be easily exploited by the attackers. By attempting to create IPSec tunnels to the router, the attacker can exhaust resources on the router.

The tunnels will never be created because authentication will fail, but resources will be allocated before the authentication failure occurs. One of the two things can happen. The router will be so busy with opening new tunnels and authentication that the forwarding of the packets will be degraded, which will constitute a denial of service. Alternatively, if there is a set limit of the number of IPSec tunnels, which the router can handle, this limit can be reached, and traceback will be impossible. For this reason, this scheme is deemed as not being able to handle complex DDos attacks.

2.3 IP Traceback State-Of-The-Art Evaluation

In this chapter, the state-of-the-art on IP traceback, along with proposed solutions to this problem have been presented. Table 2.1 provides the summary of the evaluation and offers comparison of IP traceback techniques.

As can be seen from Table 2.1, none of the methods possesses all of the qualities of the ideal scheme. Solutions to a problem are rarely ideal. Very often several solutions produce a useful taxonomy. For the problem of IP Traceback, several solutions have been proposed. Each one of them has its own advantages and disadvantages. So far, none of the methods described in this article have been used on the Internet. When economic or political incentives become strong enough to justify deployment of IP Traceback, some new requirements and metrics for evaluation might emerge.

Table 2.1 Comparison of Traceback Schemes

		PPM	iTrace	Overlay	Hash-based IP Traceback	Controlled Flooding	Traceback with IPSec
ISP Involvement		Low	Low	Large	Fair	None	High
Scalability		High	High	Poor	Fair	N/A	Poor
Vendor Involvement (# of functions to implement)		2	2	None	3	1	None
Number of Attack Packets Required for Traceback		Thousands	Thousands	1	1	Huge	Fair
Is Partial Deployment Within a Single ISP Possible?		Yes	Yes	No	Yes	N/A	Yes
Is Prior Knowledge of Topology and Routing Required for Traceback?		Yes, only if deployed partially	Yes, only if deployed partially	No	Yes, only if deployed partially	Yes	Yes
Is Inter-ISP Deployment Possible		Yes	Yes	No	Yes	Yes	Yes
Network Processing Overhead	Every Packet	Low	Low	Low	Low	None	None
	During Traceback	None	None	Low	Low	None	High
Victim Processing Overhead	Every Packet	None	None	None	None	None	None
	During Traceback	High	High	None	None	Fair	High
Bandwidth Overhead	Every Packet	None	Low	High	None	None	None
	During Traceback	None	None	None	Low	Huge	High
Memory Requirements	Network	None	Low	Low	Fair	None	None
	Victim	High	High	None	None	Low	None
Ease of Evasion		Low	High	Low	Low	N/A	Low
Protection		High	High	Fair	Fair	N/A	High
Ability to Handle Packet Transformations		Good	Good	Good	Good	Good	Good
Ability to Handle Major DDoS Attacks		Poor	Poor	Good	Good	Unable	Poor
Limitations		DoS and DDoS attacks only	DoS and DDoS attacks only	Single ISP. Single point of failure.	Strict timing constrains on traceback process. Single Point of Failure	DoS only. Manual. Unsafe. Inconsistent. Traceback is possible only while attack is in progress	Single ISP.

CHAPTER 3

BASIC DPM

The basic DPM is a packet marking algorithm, first introduced in [25]. This section provides the general principle behind DPM and discusses the most basic implementation of the proposed scheme.

3.1 Assumptions

The assumptions in this section were largely borrowed from the article by S. Savage, et al. [6]. Some of them were, however, modified to reflect the fact that the scheme is not designed merely for traceback of (D)DoS attacks.

- An attacker may generate any packet
- Attackers may be aware they are being traced
- Packets may be lost or reordered
- An attack may consist of just a few packets
- Packets of an attack may take different routes
- Routers are both Central Processing Unit (CPU) and memory limited
- Routers are not compromised

3.2 DPM Principle

As mentioned above, DPM is a packet marking algorithm. The 16-bit packet ID field and 1-bit RF in the IP header will be used to mark packets. *Each* packet is marked when it enters the network. This mark remains unchanged for as long as the packet traverses the network. This automatically removes the issue of mark spoofing which other marking schemes have to account for. The packet is marked by the interface closest to the source of the packet on an edge ingress router, as seen in Figure 3.1. The routers with the red dot signify the routers with DPM enabled, and the rubber-stamps signify the interfaces on these routers that actually perform the marking. The mark contains the partial address information of this interface, and will be addressed later in Section 3.3. The interface makes a distinction between incoming and outgoing packets. Incoming packets are marked; outgoing packets are not marked. This ensures that the egress router will not overwrite the mark in a packet placed by an ingress router.

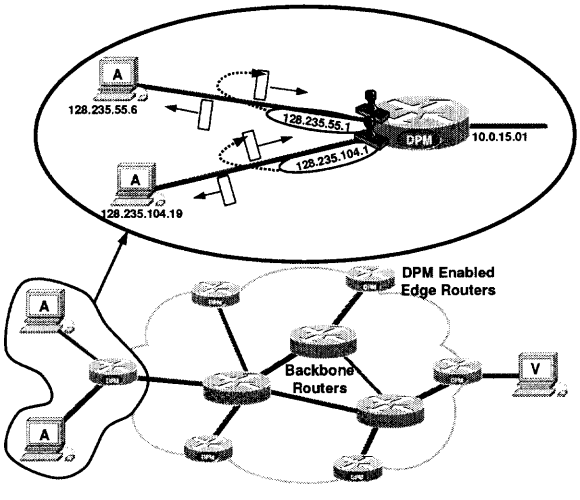

Figure 3.1 Deterministic packet marking.

For illustrative purposes, assume that the Internet is a network with a single administration. In this case, only interfaces closest to the customers on the edge routers will participate in packet marking. Every incoming packet will be marked. Should an attacker attempt to spoof the mark in order to deceive the victim, this spoofed mark will be overwritten with a correct mark by the very first router the packet traverses.

3.3 Procedure

A 32-bit IP address needs to be passed to the victim. A total of 17 bits are available to pass this information: 16-bit ID field and 1-bit RF. Clearly, a single packet would not be enough to carry the whole IP address in the available 17 bits. Therefore, it will take at least two packets to transport the whole IP address. An IP address will be split into two segments, 16 bits each: segment 0 – bits 0 through 15, and segment 1 – bits 16 through 31. The marks are prepared in advance in order to decrease the per packet processing. Each mark has two fields: Segment Number and Address bits. With probability of 0.5, the 17-bit field comprised of the ID field and RF of each incoming packet will be populated with either of those two marks.

At the victim, it is suggested that a table matching the source addresses to the ingress addresses is maintained. When a marked packet arrives to the victim, the victim will first determine if the given packet is an attack packet. If it is, the victim would check to see if the table entry for the source address of this packet already exists, and create it if it did not. Then, it would write address bits of the segment into the corresponding bits of the ingress IP address value. After both segments

```
Marking procedure at router R, edge interface A:
 for y = 0 to 1
  Marks[y].Seg_Num := y
  Marks[y].A_bits := A[y]
 for each incoming packet w
  let x be a random integer from [0,1]
  write Marks[x] into w.Mark

Ingress address reconstruction procedure at V:
 for each attack packet w
  IngressTbl[w.Mark.Seg_Num] := w.Mark.Seg_Num
```

Figure 3.2 Pseudo code for the basic DPM.

corresponding to the same ingress address have arrived to the destination, the ingress address for a given source address becomes available to the victim. The details of the procedure are shown in Figure 3.2. Both marking and reconstruction procedures are simplistic and are used in this chapter to illustrate functionality of DPM on a high level.

CHAPTER 4

MULTIPLE ATTACKERS AND IP SOURCE ADDRESS INCONSISTENCY

The limitation of the basic DPM in handling a certain type of DDoS attacks lies in the fact that the destination would associate segments of the ingress address with the source address of the attacker. If it could be guaranteed that only one host participating in the attack has a given source address, even though it might have been spoofed, and that the attacker would not change its address during the attack, the basic DPM is able to trace back. There are two situations when the reconstruction procedure of the basic DPM will fail. First, consider the situation when two hosts with the same SA attack the victim. The ingress addresses corresponding to these two attackers are A_0 and A_1, respectively. The victim would receive four address segments: $A_0[0]$, $A_0[1]$, $A_1[0]$, and $A_1[1]$. The victim, not being equipped to handle such attack would eventually reconstruct four ingress addresses, since four permutations are ultimately possible: $A_0[0].A_0[1]$, $A_0[0].A_1[1]$, $A_1[0].A_0[1]$, and $A_1[0].A_1[1]$, where '.' denotes concatenation. Only two of the four would be valid.

A typical metric of evaluation of the traceback schemes for DDoS attacks is the *rate of false positives* or *false positive rate.* In the context of DPM, a false positive is defined as an incorrectly identified ingress address. The rate of false positives refers to the ratio of the incorrectly identified ingress addresses to the total number of identified ingress addresses. In the example described above, the false positive rate for that particular attack is 50%. Clearly, the false positive rate would increase even further if the number of attackers, with the same SA, was larger.

Second, consider a (D)DoS attack, where the attackers change their source addresses for every packet they send. The basic DPM will be unable to reconstruct any valid ingress addresses since none of the entries in the *IngressTbl* would have a complete ingress address.

4.1 General Principle of Handling DDoS Attacks

A general principle in handling (D)DoS attacks of these types is to rely *only* on the information transferred in the DPM mark as was stated in [26]. The DPM Mark can be used to not only transfer the bits of the ingress address but also some other information. This additional information should enable the destination to determine which ingress address segments belong to which ingress address.

The reconstruction procedure utilizes the data structure called *Reconstruction Table* (*RecTbl*). The destination would first put the address segments in *RecTbl*, and then only after correctly identifying the ingress address, out of many possible address segments permutations, would transfer it to *IngressTbl*.

4.2 Single Digest Modification to DPM

The scheme described in this section utilizes a hash function, $H(x)$. To simplify the performance analysis, the hash function is assumed to be *ideal*. It is also assumed that the hash function is known to everybody including all DPM-enabled interfaces, all destinations which intend to utilize DPM marks for traceback, and the attackers. The constraint of 17 bits still remains, and so a longer digest would result in fewer bits of the actual address transmitted in each mark, and consequently, the higher

number of packets required for traceback. The shorter digest, on the other hand, would result few packets required to transmit all the marks, but would repeat more often for different ingress addresses.

4.2.1 Mark Encoding

Recall that in the basic DPM, the ingress address was divided into two segments. In this modified scheme, the ingress address is divided into k segments. Also, more bits would be required to identify the segment. Instead of a single bit required for two segments in the basic DPM, $log_2(k)$ would be required for this scheme. The remaining bits would be used for the digest. Independently of which segment of the address is being sent to the victim, the digest portion of the mark will always remain the same for a given DPM interface. This would enable the victim to associate the segments of the ingress address with each other to reconstruct the whole address.

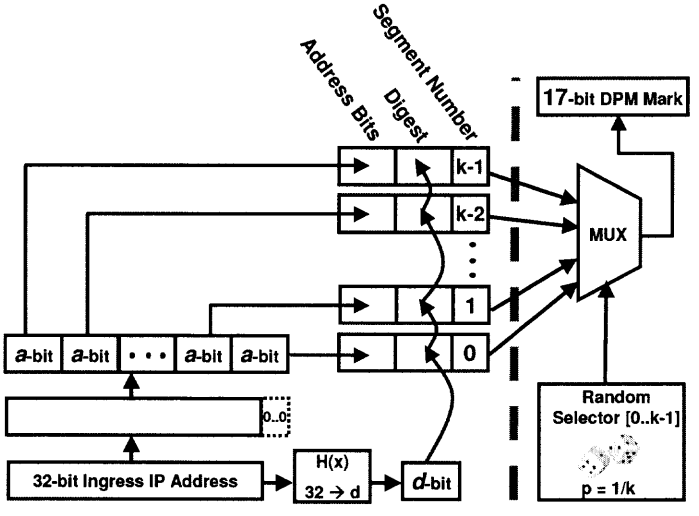

Figure 4.1 Mark encoding for single digest DDoS modification.

Figure 4.1 shows the schematics of the approach. The DPM mark consists of three fields: a-bit address segment field, d-bit digest field, and s-bit segment number field. Some padding may be required so that the address is split into segments of equal length. For example, if the ingress address is divided in five segments, it would be necessary to pad it with '000' to make it 35-bit long.

At startup the DPM-enabled interface prepares k marks for all segments of the address. A d-bit hash value, or digest, of the ingress address is calculated once and then inserted in the digest field of every mark. Each of k marks will have address bits set to a different segment of the ingress address. The segment number field will be set to the appropriate value. These operations are shown to the left of the bold dotted line in Figure 4.1. The processing required for every packet will be limited to generating a small random number from 0 to $k-1$ and inserting a corresponding mark into the packet header.

4.2.2 Reconstruction by the Victim

The reconstruction procedure of this scheme will consist of two separate processes: Mark Recording and Ingress Address Recovery. The reason for separating these two tasks is the fact that the attack packets may arrive to the destination faster than they can be analyzed. The mark recording process will set the appropriate bits in *RecTbl* to indicate which marks have arrived to the destination. Address recovery will check those bits, compose address segment permutations, and determine which ones are valid ingress addresses.

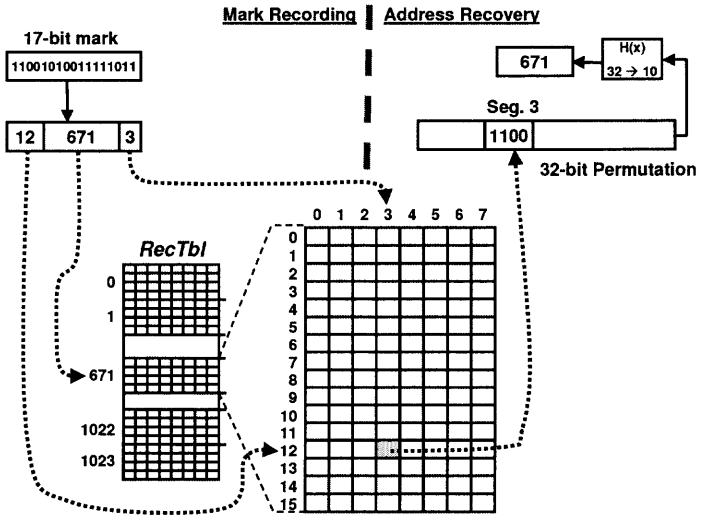

Figure 4.2 *RecTbl* with k=8, d=10, a=4; mark recording; address recovery.

A reconstruction table *RecTbl* is a 2^{17} bit structure, where every possible mark can be uniquely represented. It consists of 2^d areas. Each area consists of k segments, and each segment consists of 2^a bits. Figure 4.2 shows an example of *RecTbl*, where k, d, and a are 8, 10 and 4, respectively. When a mark becomes available to the mark recording process, it sets the appropriate bit in the *RecTbl*. For a given attacker, the ingress address can possibly hashed into 2^d digest values. The digest is extracted from the mark and the area where the bit will be set is determined. The segment number field in the mark indicates the segment in the *RecTbl* area, where the appropriate bit would be set. Finally, the value of the address bits in the mark indicates the actual bit, which will be set to '1'. This process is repeated for every mark.

The address recovery process will be a part of a larger traceback procedure. It will analyze each area of the *RecTbl*. Once again, it runs independently from the mark recording process, thus allowing post-mortem traceback. The value of a certain

bit in *RecTbl* indicates if the corresponding mark arrived to the victim. For example, bit 12 in segment 3 of area 671 set to '1' means that there is an ingress address of interest, with digest of 671 having segment 3 equal to '1100'$_2$ as shown in Figure 4.2. This segment has to be combined with other segments of this area in order to create permutations of segments. Hash function, $H(x)$, is applied to each of these permutations. If the result matches the area number, which is actually the digest embedded in the marks (in this example 671), then the recovery process concludes that this permutation of segments is in fact a valid ingress address. The process moves on to the other segments in this area of *RecTbl*. The address recovery process has to go through all permutations within a given area before moving on to another area. $H(x)$ is applied to every permutation, and only if the result matches the area number, the permutation is transferred to the *IngressTbl*. Additionally, if there is a segment in the area, which has no bits asserted, then the process can immediately move on to another area of *RecTbl*. Details of the mark encoding procedure, the mark recording and the address recovery processes for the single digest modification are shown in Figure 4.3.

4.2.3 Analysis

In this section, the number of attackers, which this modified scheme can traceback, with the false positive rate limited to 1%, is evaluated. Let us examine the origin of false positives. If there is only one ingress address with a given digest, there will be no false positives; however, as N increases, the chance of the digest repeated for another address also increases. The expected number of digests for a certain number

of N can be thought of as the expected number of the faces turning up on a 2^d-sided die after N throws. This is a special case of a classical occupancy problem discussed by W. Feller [27]. The expected number of different digests, $E[H]$, is:

$$E[H] = 2^d - 2^d \left(1 - \frac{1}{2^d}\right)^N . \tag{4.1}$$

Therefore, the rate of false positives is 0 for the values of N, for which the expected number of digests, $E[H]$, equals to N, since every ingress address will have a unique digest.

Since there may be more than one address resulting in the same digest, each segment associated with a given digest would have a certain number of values. For example, if two addresses have the same digest, segment 0 in the area of the *RecTbl* corresponding to this digest could have either one or two bits set to '1'. If segment 0 in these two addresses is the same, then there would be only one bit set to '1', and if segment 0 of one address is different from segment 0 of the second address, then two bits will be set to '1'. The expected number of values that a segment will assume can also be thought of as the expected number of the faces turning up on a 2^a-sided die after N_d throws, as shown in [27], where N_d is the number of ingress addresses with the same digest. The expected number of different values the segment will take is

$$2^a - 2^a \left(1 - \frac{1}{2^a}\right)^{N_d}, \tag{4.2}$$

for those areas, which have segments of more than one ingress addresses, and 1 for those which have segments of only a single ingress address. The expected number of all permutations of address segments for a given digest is

$$\left[2^a - 2^a\left(1 - \frac{1}{2^a}\right)^{N_d}\right]^k.$$

Recall that after a permutation of segments is obtained, the hash function $H(x)$ is applied to it, and if the result does not match the original digest, that permutation is not considered. The expected number of permutations that result in a given digest for a given area of the *RecTbl* is

$$\frac{\left[2^a - 2^a\left(1 - \frac{1}{2^a}\right)^{N_d}\right]^k}{2^d}.$$

The number of false positives for a given area would be the total number of permutations, less the number of valid ingress addresses, which match the digest. For this modification, just a few areas, which have segments of more than one ingress addresses, will produce more than $0.01N$ of false positives. It is assumed that for all those areas $N_d = 2$. The number of those areas is $N - E[H]$, and the number of valid ingress addresses with segments in those areas is $2(N - E[H])$. The number of false positives is given by

$$\frac{(N - E[H])\left[2^a - 2^a\left(1 - \frac{1}{2^a}\right)^2\right]^k - 2(N - E[H])}{2^d} \tag{4.3}$$

This number has to be less than 1% of N. Therefore, Eq. (4.3) has to be set to be less or equal to $0.01N$, and solved for N. Recall that a, d, and $E[H]$ can be expressed in

terms of k. The maximum N, which would satisfy this inequality, N_{MAX}, is difficult to be expressed in terms of k. However, it is possible to find N_{MAX} by substitution. Table 4.1 provides the values of N_{MAX} for selected values of k.

Another important consideration is the expected number of datagrams required for the reconstruction. This number is related to k, the number of segments that the ingress address was split. The larger the k, the more different packets it would be required for the victim to receive in order to reconstruct the ingress address. The expected number of datagrams, $E[D]$, required to be marked by a single DPM-enabled interface in order for the victim to be able to reconstruct its ingress address is given by a Coupon Collector problem also described by W. Feller [27]:

$$E[D] = k\left(\frac{1}{k} + \frac{1}{k-1} + \ldots + 1\right).$$

Table 4.1 provides the value of $E[D]$ for selected values of k.

Table 4.1 Relationship Between Selected k and a, s, d, N_{MAX}, and $E[D]$ for the Single Digest Modification

k	*a*	*s*	*d*	N_{MAX}	*E[D]*
2	16	1	0	1	2
4	8	2	7	26	8
8	4	3	10	108	22
16	2	4	11	45	55
32	1	5	11	45	130

```
Marking procedure at router R, edge interface A:
  Digest:= H(A)
  for y = 0 to k − 1
    Marks[y].Digest := Digest
    Marks[y].Seg_Num := y
    Marks[y].A_bits := A[y]
  for each incoming packet w
    let x be a random integer from [0,k)
    write Marks[x] into w.Mark


Mark Recording process at victim V:
  for each attack packet w
     Area := w.Mark.Digest
     Seg := w.Mark.Seg_Num
     Bit := w.Mark.A_bits
     RecTbl[Area, Seg, Bit] := '1'

Address Recovery process at victim V:
  for Area = 0 to 2^d − 1
    for Bit_0 = 0 to 2^a − 1
      if RecTbl[Area, 0, Bit_0] == '1' then
      ...
        if RecTbl[Area, k − 2, Bit_{k−2}] == '1' then
          for Bit_{k−1} = 0 to 2^a − 1
            if RecTbl[Area, k − 1, Bit_{k−1}] == '1' then
              Prm := Bit_0 . Bit_1 . .... . Bit_{k−1}
              Digest := H(Prm)
              if Area == Digest then
                Prm ⇒ IngressTbl
```

Figure 4.3 Pseudo code for the modified single digest DPM algorithm.

4.3 Multiple Digest DDoS Modification to DPM

In the scheme described in Section 4.2, a single hash function, $H(x)$, was used for identifying the segments of the same ingress address. In this section, a modification, requiring a family of hash functions, is introduced.

4.3.1 Mark Encoding

In this scheme, the family of f hash functions, $H_0(x)$ – $H_{f-1}(x)$, will be used to produce f digests of the ingress address. As in the single digest scheme described in Section 4.2.1, the address segment and the segment number will be transferred in each mark. Instead of the single digest, however, one of the several digests produced by each of f hash functions concatenated with the function identifier will be embedded in the mark. The d-bit field, which was used solely for the digest in the single-digest scheme, would have to be split into two fields. One field, $log_2(f)$-bit long carrying the identifier of the hash function, and d-bit field with the digest itself.

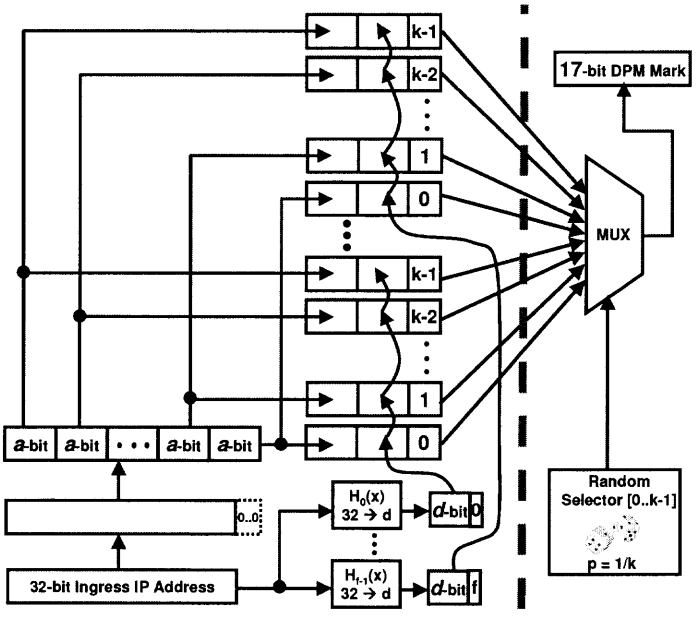

Figure 4.4 Mark encoding for multiple digest DDoS modification.

Figure 4.4 illustrates the process of the mark encoding. The process is very similar to the one described in 4.2.1, but differs in that for every ingress address, not k, but $f \times k$ marks have to be created at startup and then randomly selected for every packet. That does not affect the DPM-enabled interface per-packet overhead, since it

will be limited to generating a small random number and overwriting 17 bits in the header, just as for the single-digest or basic DPM schemes. The one-time penalty of calculating the digests and producing the marks are is irrelevant.

4.3.2 Reconstruction by the Destination

Reconstruction by the destination is also similar to that described in Section 4.2.2. The structure of *RecTbl* has to be changed slightly. The *RecTbl* will consist of f smaller parts. Every one of those parts will have the structure identical to the *RecTbl* described in Section 4.2.2 (2^d areas, k segments in every area, and 2^a bits in every segment). The mark recording process first examines the hash function identifier field. Then it proceeds to the corresponding part of the *RecTbl*. Having identified the part in the *RecTbl*, the area, and the segment, the appropriate bit is set to '1', as in the single-digest scheme.

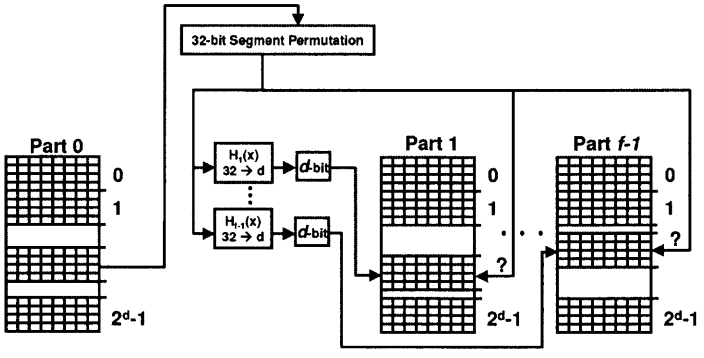

Figure 4.5 Address recovery for multiple digest DDoS modification.

The address recovery process, shown in Figure 4.5, identifies the permutations which match the digest in areas of *Part*0 of *RecTbl*. Once a permutation is validated by comparing its digest obtained by applying $H_0(x)$ to the area number, the rest of the hash functions, $H_1(x)$ to $H_{f-1}(x)$, are applied to it to produce $f-1$ digests. These

digests are used to verify the existence of this permutation in other parts of *RecTbl*. The process then checks these areas of the remaining parts for the permutation in question. If the permutation is present in the appropriate area of every part of the *RecTbl*, it is concluded that the permutation is a valid ingress address. Notice that the permutation does not have to be verified in every part. It is known that the digest obtained by applying $H_i(x)$ to the permutation being checked will match the area number since the area was identified by this operation. Therefore, such verification would be redundant and will always produce a positive outcome. The pseudo code in Figure 4.6 provides the details of the mark encoding procedure, mark recording, and address recovery processes.

4.3.3 Analysis

The purpose of the analysis of this scheme remains the same: to find N_{MAX}, the maximum number of simultaneous attackers, which can be traced back with the false positive rate not exceeding 1%. For the multiple digest scheme, the number of false positives in one area of *RecTbl* can be higher than in a single digest scheme because the same false positive has to appear in the appropriate areas of all other parts of *RecTbl* in order to be identified as an ingress address.

Recall, from Section 4.2.3, that the expected number of permutations in a given area is given by

$$\left[2^a - 2^a\left(1 - \frac{1}{2^a}\right)^{N_d}\right]^k,$$

where N_d is the number of ingress address with this digest. Since for the multiple digest scheme, unlike the single digest scheme, the number of ingress addresses with the same digest will be more than 2, the following analysis is more suitable. The number of ingress addresses with the same digest is $\frac{N}{E[H]}$. The number of permutations in a single digest is then

$$\left[2^a - 2^a\left(1 - \frac{1}{2^a}\right)^{\frac{N}{E[H]}}\right]^k.$$

The number of false positives for this digest is

$$\frac{\left[2^a - 2^a\left(1 - \frac{1}{2^a}\right)^{\frac{N}{E[H]}}\right]^k - N}{2^d}.$$

The number of false positives in $Part0$ is given by:

$$\frac{E[H]}{2^d}\left(\left[2^a - 2^a\left(1 - \frac{1}{2^a}\right)^{\frac{N}{E[H]}}\right]^k - N\right).$$

For large values of N, $E[H] = 2^d$, and thus $\frac{E[H]}{2^d} = 1$. So the number of false positives in $Part0$ is

$$\left[2^a - 2^a\left(1 - \frac{1}{2^a}\right)^{\frac{N}{E[H]}}\right]^k - N. \tag{4.4}$$

Once the permutation was identified as a possible ingress address in $Part0$, the remaining digests are calculated. Since the uniform distribution of addresses is assumed, any permutation is as likely to appear as any other. The probability of

any random permutation to appear is $\frac{1}{2^{32}}$. The probability that a given permutation, which is a false positive, will occur in the appropriate area of $Part1$ is:

$$\frac{\left[2^a - 2^a\left(1 - \frac{1}{2^a}\right)^{\frac{N}{E[H]}}\right]^k}{2^{32}}$$

Note that this expression is not divided by 2^d since, if the permutation in question is present in the identified areas of all other parts, it must match the appropriate digest per discussion at the end of Section 4.3.2. The probability that a given permutation will occur in the appropriate areas of all parts of *RecTbl* is:

$$\left[\frac{\left[2^a - 2^a\left(1 - \frac{1}{2^a}\right)^{\frac{N}{E[H]}}\right]^k}{2^{32}}\right]^{f-1}$$

Multiplying this expression by the number of false positives in $Part0$ results in the number of false positives after areas matching the digests 1 through $f-1$ in all the other parts of the *RecTbl* were checked. This is the total number of false positives for the *RecTbl*. Setting it not to exceed $\frac{N}{100}$ results in the following inequality:

$$\frac{\left\{\left[2^a - 2^a\left(1 - \frac{1}{2^a}\right)^{\frac{N}{E[H]}}\right]^k\right\}^f}{2^{32(f-1)}} \leq \frac{N}{100}$$

Recall that a, d, and $E[H]$ can be expressed in terms of k. So the whole inequality can be expressed in terms of k and f. Similar to the single-digest scheme, N_{MAX} can be found by substitution.

The expected number of datagrams required to reconstruct the ingress address is now given by

$$E[D] = f \times k \left(\frac{1}{f \times k} + \frac{1}{f \times k - 1} + \ldots + 1 \right).$$

Table 4.2 provides the values of N_{MAX} and $E[D]$ for selected combinations of f, a, k, and d.

Table 4.2 Relationship Between f, a, k, s, d, N_{MAX}, and $E[D]$ for Selected Combinations for Multiple Digest Modification

f	*k*	*a*	*d*	N_{MAX}	*E[D]*
4	8	4	8	2911	130
4	4	8	5	2299	55
8	4	8	4	2479	130

Multiple digest modification enables DPM to trace hundreds and even thousands of simultaneous attackers with spoofed SAs, while the $E[D]$ is either slightly increased or not increased at all, depending on the value of f and k compared to the single digest modification.

Marking procedure at router R, edge interface A:
for $z = 0$ **to** $f - 1$
  $Digest := H_z(A)$
  for $y = 0$ **to** $k - 1$
    $Marks[z \times k + y]$.Hash_num $:= z$
    $Marks[z \times k + y]$.Digest $:= Digest$
    $Marks[z \times k + y]$.Seg_Num $:= y$
    $Marks[z \times k + y]$.A_bits $:= A[y]$
for each incoming packet w
  let x be a random integer from $[0, f \times k)$
  write $Marks[x]$ into w.Mark

Mark Recording process at victim V:
for each attack packet w
  $Part := w$.Mark.Hash_num
  $Area := w$.Mark.Digest
  $Seg := w$.Mark.Seg_Num
  $Bit := w$.Mark.A_bits
  $RecTbl[Part, Area, Seg, Bit] :=$ '1'

Address Recovery process at victim V:
for $Area = 0$ **to** $2^d - 1$
  for $Bit_0 = 0$ **to** $2^a - 1$
    if $RecTbl[0, Area, 0, Bit_0] ==$ '1' **then**
    ...
      if $RecTbl[0, Area, k-2, Bit_{k-2}] ==$ '1' **then**
        for $Bit_{k-1} = 0$ **to** $2^a - 1$
          if $RecTbl[0, Area, k-1, Bit_{k-1}] ==$ '1' **then**
            $Prm := Bit_0 \,.\, Bit_1 \,.\, \ldots \,.\, Bit_{k-1}$
            $Digest := H_0(Prm)$
            if $Area == Digest$ **then**
              for $Part = 0$ **to** $f - 1$
                for $Seg = 0$ **to** $k - 1$
                  if $RecTbl[Part, H_{Part}(Prm), Seg, Bit_{Seg}] \neq$ '1' **then**
                    $False\_flag :=$ '1'
              if $False\_flag \neq$ '1' **then**
                $Prm \Rightarrow IngressTbl$

Figure 4.6 Pseudo code for the modified multiple digest DPM algorithm.

CHAPTER 5

ACCOMMODATING FRAGMENTATION

Fragmented traffic constitutes, between 0.25% according to the article by S. Savage et al. [6] and 0.5% of the total IP traffic according to the article by C. Shannon [28]. Though the amount of fragmented traffic is small, it does exist. The DPM scheme, discussed so far, did not differentiate between fragmented and non-fragmented traffic. The ID Field, which is used for fragmentation, and RF of the IP header are completely replaced with one of $f \times k$ marks chosen at random in every packet.

In this chapter, the issue of why DPM is a poor way to handle fragmented traffic is explored, and modifications to DPM to address fragmentation are presented.

5.1 IP Fragmentation Background and Terminology

Terminology used to describe different aspects of fragmentation is largely adopted from [28].

Fragmentation is a feature of IP to enable transport of packets across the networks with different Maximum Transfer Unit (MTU). *Path MTU* is the smallest MTU of all the links on a path from a source host to a destination host as described in Request for Comments (RFC) 1191 [29]. When a packet enters a network, whose MTU is smaller than the packet length, the packet has to undergo a process of *fragmentation.*

Figure 5.1 illustrates this process and introduces several important terms. *Original datagram* is an IP datagram that will be fragmented because its size exceeds the MTU of the next link. A *Packet Fragment*, or simply a *fragment*, refers to a packet

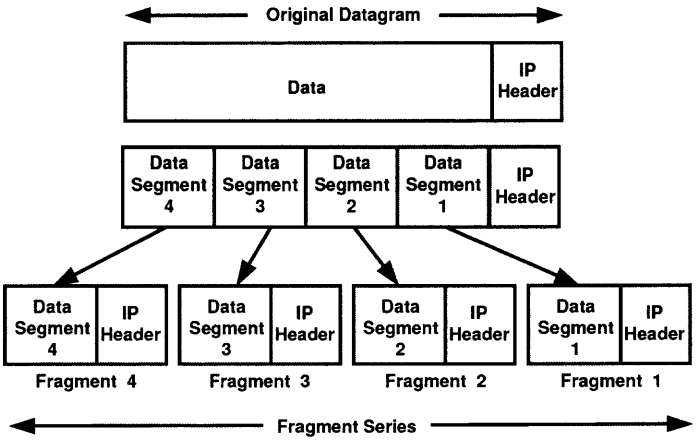

Figure 5.1 IP fragmentation.

containing a portion of the payload of an original datagram. While the datagram and packet are synonymous, the terms, *original datagram* and *packet fragment*, will be used for clarity. A *fragment series*, or simply a *series*, is an ordered collection of fragments that results from a single original datagram.

When fragmentation occurs, each fragment becomes a valid IP packet. All the fragments have their own IP header. Most of the fields of the IP header of the fragments are inherited from the original datagram IP header. The fields of interest are ID field, Flags, and Offset. ID field is copied from the original datagram to all the fragments. The SA, Destination Address (DA), Protocol (P), and ID, are used by the destination to distinguish the fragments of different series according to RFC 791 and 815 [30], [31]. The ID field of all the fragments, which resulted from a single datagram, must have their ID field in the IP header set to the same value for proper reassembly. More Fragments (MF) flag is set to '1' in every fragment except the last one. This flag indicates that more fragments to follow. The last fragment has MF set to '0' to indicate that it is the last fragment in the series. Finally, the offset field of

the IP header is set to the position of the data in the fragment with respect to the beginning of data in the original datagram. The unit of offset is eight bytes.

For successful reassembly, the destination has to acquire all of the fragments of the original datagram. A tuple (SA, DA, P, ID) is used to determine if the fragments belong to the same original datagram, MF is used to indicate the number of fragments, and Offset is used to determine the correct order of reassembly. Notice that the fragments may come out of order but reassembly will still be successful because the destination would be able to determine that the fragment belongs to a given series, and its position relative to other fragments.

Since DPM uses the ID field for its purposes, the reassembly errors at the destination may occur. First the effects of the basic DPM on reassembly are examined and then the techniques to avoid the undesirable effects are introduced. The performance of the techniques is analyzed in terms of the probability of reassembly error.

5.2 Shortcomings of DPM related to Fragmentation

Fragmentation can happen *upstream* or *downstream* from the point of marking according to S. Savage [6]. These two situations have to be considered separately.

5.2.1 Upstream Fragmentation

Upstream fragmentation is known to the DPM-enabled interface. The DPM-enabled interface can identify a packet to be a fragment by examining its MF and Offset.

In the case of upstream fragmentation, a datagram is fragmented by a router or a host before it reaches the DPM-enabled interface. When a series of fragments

of the original datagram reaches the DPM-enabled interface the ID and RF fields of all the fragments will be replaced with one of the $f \times k$ marks *picked at random.* This will cause fragments to have different ID fields when they arrive to the destination. Fragments with different ID fields will be considered to be parts of different datagrams. The reassembly will eventually timeout since the destination will never get all the fragments necessary for the reassembly of what it considers to be two separate series. The probability of all fragments in a series of two fragments having the same ID field after marking is $\frac{1}{fk}$. For a series of three packets, $\frac{1}{(fk)^2}$, etc. For $f \times k = 16$, the probability of a series consisting of two fragments being correctly reassembled is 6.25%, for a series of three fragments – 0.4%. Clearly, the rate of reassembly errors caused by upstream fragmentation is unacceptable. The ability of DPM-enabled interface to recognize upstream fragmentation results in a different strategy for marking these packets described in Section 5.3.

5.2.2 Downstream Fragmentation

Downstream fragmentation is unknown to DPM. The DPM-enabled interface has no knowledge if the marked datagrams, are being fragmented anywhere along the path. Therefore, the datagrams, which will be fragmented after the marking cannot be treated differently from the traffic, which is not fragmented.

Luckily, fragmentation downstream from the DPM-enabled interface does not causes any problems for reassembly. The router, which is going to perform fragmentation, will simply insert the content of the ID field of the original datagram into every fragment. The value of RF will also be copied to every fragment as specified

in RFC 1812 [32]. At the destination, reassembly will be successful since the ID field will be the same for every fragment in the series. The fact that the ID field was set by DPM, and replaced the original value set by the host is unknown to the destination, and is irrelevant for the purpose of reassembly.

5.3 Fragment-Persistent DPM

In this section, the modification to the DPM marking procedure which would eliminate most of the potential errors associated with upstream fragmentation is introduced. The fundamental modification will be discussed first, followed by gradual changes resulting in the final marking procedure.

5.3.1 Fundamentals of Handling Upstream Fragmentation with DPM

It is essential for proper reassembly that all of the fragments of the original datagram have the same ID field. The basic DPM marks packets randomly choosing among $f \times k$ marks. This randomness must be suspended when processing fragments. In order to accomplish this task, DPM has to keep track of the fragments which pass through. If a certain mark was inserted in the first fragment, that DPM-enabled interface encounters (which does not have to be the fragment with offset 0), then the same mark must be inserted into the rest of the fragments of this series. The information about which mark is used for which series has to be stored in a table, called $FragTbl$, at the DPM enabled interface and checked every time a new fragment arrives. To identify fragments belonging to the same original datagram, DPM should check if the tuple of the four fields utilized by the reassembly function (SA, DA, P,

ID) is the same as any other it marked within the maximum reassembly timeout of 120 seconds.

```
Marking procedure at router R, edge interface A:
 for z = 0 to f − 1
  Digest:= H_z(A)
  for y = 0 to k − 1
   Marks[z × k + y].Hash_num := z
   Marks[z × k + y].Digest := Digest
   Marks[z × k + y].Seg_Num := y
   Marks[z × k + y].A_bits := A[y]
 for each incoming packet w
  let x be a random integer from [0, f × k)
   if w.MF == '1' OR w.offset ≠ 0 then
    if FragTbl[SA, DA, P, ID] == NIL then
     create FragTbl[SA, DA, P, ID]
     FragTbl[SA, DA, P, ID] := x
    else
     x := FragTbl[SA, DA, P, ID]
  write Marks[x] into w.Mark
```

Figure 5.2 Pseudo code for the fragment-persistent DPM.

Figure 5.2 illustrates the fundamental changes to the DPM marking procedure for fragmentation support. If the packet is not a fragment, there would be no changes in handling it. If, however, the packet is a fragment, then DPM determines if it is the first fragment in the series that it sees. If it is the first one, then the process is identical to the non-fragment case, but, in addition, DPM stores the result of the concatenated hash function number and segment number. This $log_2(f \times k)$-bit pattern uniquely identifies every mark at this interface, and is arithmetically equal to the index of the *Marks* array used in the procedure. This would allow to set the mark of all the remaining fragments in this series to the same value as the first fragment. When the packet is identified as a fragment and DPM marking procedure was able

to find the hash function number and the segment number assigned to its series, the corresponding mark is inserted. The reconstruction procedure at the victim will not change and will be identical to the reconstruction procedure of the basic DPM.

5.3.2 Dealing with Infinite Series

Assuming that an attacker can generate any packet, it is possible that he will utilize *artificial fragmentation.* That is sending packets with MF Flag set to '1' or non-zero offset field when fragmentation is not necessary for the proper reason – the datagram exceeding the MTU of a given link. With artificial fragmentation, the attacker may generate infinitely many packets with the same SA, DA, P, and ID fields, that would look like fragments of one very long series to the DPM-enable interface or the destination. This is known as an *infinite series.* The invalid traffic would be noticed only by the destination at the reassembly function, but for (D)DoS attacks it would be enough that the invalid packets occupy the resources of the victim. In this situation, the victim will never recover the full ingress address since only a single mark would be available.

To remedy this situation, another simple modification in addition to fragment persistence must be introduced. The modification is based on the findings in [28], where it was determined from the real traffic traces that the longest series on the Internet is 44 fragments. Deterministic Packet Marking should recognize the fact that if the number of fragments in the series exceeds 44, it is, in all likelihood, an attack, or a result of some errors. In either case, such traffic is not expected to be properly reassembled. So, after DPM has persistently marked 44 fragments of a

single series with the same mark, any additional fragments from the same series will be marked randomly, as if it was not a fragment.

In order to implement this modification, the *FragTbl*, which DPM uses to account for fragments and where the segment value corresponding to (SA, DA, P, ID) is kept, should also keep a counter, which should be incremented every time a fragment with a given tuple is encountered. Once this counter exceeds 44, marking persistence should be suspended and randomness should be reinstated. Figure 5.3 illustrates this concept with a pseudo code.

```
Marking procedure at router R, edge interface A:
 for z = 0 to f − 1
  Digest:= H_z(A)
  for y = 0 to k − 1
   Marks[z × k + y].Hash_num := z
   Marks[z × k + y].Digest := Digest
   Marks[z × k + y].Seg_Num := y
   Marks[z × k + y].A_bits := A[y]
 for each incoming packet w
  let x be a random integer from [0, f × k)
   if w.MF == '1' OR w.offset ≠ 0 then
    if FragTbl[SA, DA, P, ID] == NIL then
     create FragTbl[SA, DA, P, ID]
     FragTbl[SA, DA, P, ID].Mark_num := x
     FragTbl[SA, DA, P, ID].counter := 1
    else
     if FragTbl[SA, DA, P, ID].counter < 45 then
      x := FragTbl[SA, DA, P, ID].Mark_num
      FragTbl[SA, DA, P, ID].counter++
  write Marks[x] into w.Mark
```

Figure 5.3 Pseudo code for the fragment-persistent DPM with fragment counter.

5.3.3 Practical Compromise

The modification described in Section 5.3.2 will accommodate all of the valid fragmented traffic. However, artificial fragmentation may still be used by the attacker to generate bogus 44-fragment series directed to the victim. This will allow the attacker to increase the expected number of packets required to be marked by a DPM-enabled interface in order for the victim to be able to reconstruct its address, $E[Pkt]$, by the factor of 44. It is possible to modify the procedure outlined in Section 5.3.2 to significantly reduce this factor with the minimal trade-off.

According to C. Shannon [28], about 99% of series are only two or three fragments long. This fact may be taken into consideration when resuming randomness. It follows then that if the randomness in selecting the mark is resumed after only three fragments have passed through the DPM-enabled interface, 99% of fragmented datagrams will be unaffected and will reassemble successfully at the destination. To the attacker, this will make sending series longer than three fragments to the victim totally pointless. For example sending a series of 45 fragments will result in three fragments marked with the same mark, and the remaining 42 fragments marked randomly. The marks will be picked at random 43 times. Assuming $f \times k = 16$, approximately 15 different marks will be sent to the victim, according to the classical occupancy problem discussed in [27]. The same number of packets may be sent to the victim if the attacker sends 15 series, three fragments each. All three fragments in every series will be marked with the same mark. Therefore, random mark will be picked only 15 times, resulting in approximately 10 different marks sent to the victim. Clearly, sending series of three fragments to the victim becomes the most

sensible option for the attacker. While this approach will take care of all two and three fragment series, which account for 99% of all series, the remaining 1% of valid series, which contain more than three fragments will almost never get reassembled at the destination.

The compromise approach to the fragmentation problem is now presented. When the DPM-enabled interface encounters the first (not necessarily with offset 0) fragment in a series, it decides if the randomness will be suspended for three fragments or for 44 fragments in this series. The probability p, with which the randomness is suspended for 44 fragments, should be selected in such a way that there is no advantage to the attacker in sending series longer than three fragments.

Sending series of more than 44 fragments does not make any sense. It is certain that the marks selected at random will be inserted in the fragments after the 44th. However, the attacker may send series of exactly 44 fragments hoping that the number of packets sent to the victim would be greater than it were using three fragment series for the same number of marks. If the attacker generates 44 fragment series, the situation when only a single mark is inserted in all the fragments will occur with probability p. The alternative is the situation when only the first thee fragments will have the same mark, and the remaining 41 fragments will have randomly picked marks inserted. Thus, 42 randomly picked marks would be transferred to the victim in the fragments of this series. This situation will occur with probability $(1-p)$.

It is desired to find the value of p such that the expected number of packets per randomly selected mark, C, is the same for both approaches. This would minimize the undesirable effect the modification has on longer valid series without creating any

benefit to the attacker of using the longer artificial series. Denote D as the number of datagrams being sent. In case of sending series of three fragments, the expected number of times marks are randomly picked (different from the number of marks acquired by the victim) is D, and the number of packets sent to the victim is $3D$. In case of sending 44 fragment series, the expected number of randomly chosen marks is $D(42(1-p)+p)$, and the number of packets sent to the victim is $44D$. The ratio of number of packets to the number of generated marks will be called a fragmentation coefficient C. For the two options of using artificial fragmentation, C must be the same.

$$\frac{44D}{D(42(1-p)+p)} = \frac{3D}{D}$$
$$\frac{44}{42(1-p)+p} = 3$$

Solving for p results in the value of $\frac{2}{3}$. It is important that the number of datagrams sent by the given host does not affect the value of p. This means that DPM can suspend randomness in mark selection for 44 fragments in two out of every three datagrams. Approximately 33.3% of the datagrams fragmented into more than three fragments upstream would fail to reassemble at the destination. The fragmented traffic is only 0.5% of the overall traffic. Therefore about 0.0017% of the overall traffic would be affected. The pseudo code of the encoding procedure reflecting the practical compromise is depicted in Figure 5.4. Processing at the victim is not affected by any of these modificatinos.

```
Marking procedure at router R, edge interface A:
 for z = 0 to f − 1
  Digest:= H_z(A)
  for y = 0 to k − 1
   Marks[z × k + y].Hash_num := z
   Marks[z × k + y].Digest := Digest
   Marks[z × k + y].Seg_Num := y
   Marks[z × k + y].A_bits := A[y]
 for each incoming packet w
  let x be a random integer from [0, f × k)
  if w.MF == '1' OR w.offset ≠ 0 then
   if FragTbl[SA, DA, P, ID] == NIL then
    create FragTbl[SA, DA, P, ID]
    let v be a random integer from set {3, 44, 44}
    FragTbl[SA, DA, P, ID].Mark_num := x
    FragTbl[SA, DA, P, ID].counter := 1
    FragTbl[SA, DA, P, ID].Suspend := v
   else
    if (FragTbl[SA, DA, P, ID].counter
    ↪   < FragTbl[SA, DA, P, ID].Suspend) then
     x := FragTbl[SA, DA, P, ID].Mark_num
     FragTbl[SA, DA, P, ID].counter++
  write Marks[x] into w.Mark
```

Figure 5.4 Pseudo code for the practial compromise fragment-persistent DPM.

5.4 Size of the *FragTbl*

In this section, the amount of memory required for the *FragTbl* is analyzed. This is an important issue since this memory overhead will be incurred by the routers, and as was mentioned earlier the ISPs involvement for the scheme should be minimal. The amount of memory required for the *FragTbl* depends on the interface speed and will vary for different interfaces. In this section, the estimation of the size of *FragTbl* is presented.

The size of *FragTbl* is directly proportional to the rate of the DPM enabled interface, R. The interfaces with the higher rate are able to process more packets per second. As mentioned earler, according to C. Shannon [28], approximately 0.5% of IP packets are fragmented. For every series, 12 Bytes (4-Byte SA, 4-Byte DA, 2-Byte ID, 1-Byte P, 4-bit fk value, and 1-bit required to store two values of threshold for the number of fragments to resume randomness) are allocated in the *FragTbl* and every entry should be held in the *FragTbl* for 120 seconds. Keeping the entry longer than 120 seconds is unpractical since the reassembly process at the destination of the fragments will timeout after 120 seconds according to RFC 1122 [33]. The average packet size of 1000 bits is conservatively considered as it was by A. Snoeren [19]. The recent traffic measurement studies suggest that the average packet size is, however is closer to 400 to 600 Bytes according to S. McCreary and K. Claffy [34], and S. Bhattacharyya [35]. It follows then that the size of the *FragTbl* in Bytes is given by:

$$\frac{R\,bits/s \times 120\,s \times 0.005 \times 12\,Bytes}{1000\,bits} = 0.0072R\,MBytes$$

Table 5.1 summarizes memory requirements of *FragTbl* for various commonly used interfaces. The interfaces, which are likely to be on the edges of even a large ISP, would not require more than 20MBytes of Random Access Memory (RAM).

Table 5.1 Interfaces, Rates and Estimated *FragTbl* Size

Interface	Rate	*FragTbl* Size
OC-768	40Gb/s	288 MByte
OC-192, 10GigE	10Gb/s	72 MByte
OC-48	2.5Gb/s	18 MByte
OC-24	1.25Gb/s	9 MByte
GigE	1Gb/s	7.2 MByte
OC-12	622Mb/s	4.5 MByte
OC-3	155Mb/s	1.12 MByte
Fast Ethernet	100Mb/s	0.72 MByte
OC-1	51.84Mb/s	0.37 MByte
DS3	44.736Mb/s	0.33 MByte
DS2	6.312 Mb/s	< 64 KByte
DS1	1.544 Mb/s	
DS1C	3.152 Mb/s	
DS0	64 kb/s	

CHAPTER 6

TRACEBACK

In this chapter, the different types of possible attacks to which the victim may be subjected are discussed and the traceback procedure designed to perform the traceback for all types of attacks is introduced. Then the conditions for traceability for each attack type are discussed. Finally, the simulation results are presented and discussed.

6.1 Types of Cyber Attacks

E. Carter [36] divides the attack signatures into four classes: reconnaissance, informational, access, and denial of service. The first three classes can be combined into one in the context of the DPM traceback and will be called *intrusions*. The last class is the most challenging in terms of traceback, and will be discussed separately.

6.1.1 Intrusions

The important characteristic of an intrusion is that the attacker is interested to receive some information from the victim. The attacker is thus restricted to use a stable IP address in order to receive the replies from the victim. Even if the attacker engages in an elaborate scheme when his/her address is spoofed, but he/she is still capable of receiving the packets from the victim, and thus the traceback of such attacks is still trivial.

6.1.2 Denial of Service Attacks

Denial-of-service attacks have become very popular recently. Currently, there is no complete comprehensive defense against these attacks. That is why the traceback of these attacks becomes even more important. The common goal of all denial-of-service attacks is to create a situation when the victim is unable to provide services to the customers. This is usually accomplished by exhausting the physical or logical resources on the victim's servers and networks or the ISP uplink.

A required attribute of any DDoS attack is a collection of *slaves*. The slaves are the hosts on the Internet that the attacker compromises by using common vulnerabilities and bugs of the operating systems as stated by V. Paxson in [37]. When the attacker compromises a slave and gains full or partial control, the flood servers are installed in them. Since the attacker controls the slaves, it is possible to have the slaves generate any packet. Packets with the spoofed SA and artificial fragmentation are of particular interest. The DDoS attack may also involve *reflectors*, the uncompromised hosts with opened services (such as www), which are used to reflect the traffic from the slaves to the victim. The mechanism of reflections works as follows. The slaves forge the source address in the packet directed to the reflectors for the victim's IP address. The result is that the victim is being flooded by the replies, which were originated by the innocent servers. A feature of reflectors is that the attacker has no control over the them, and therefore only valid packets may be generated by a reflector. That means the SA field will have the reflector's address, and the artificial fragmentation is not orchestrated by the reflectors.

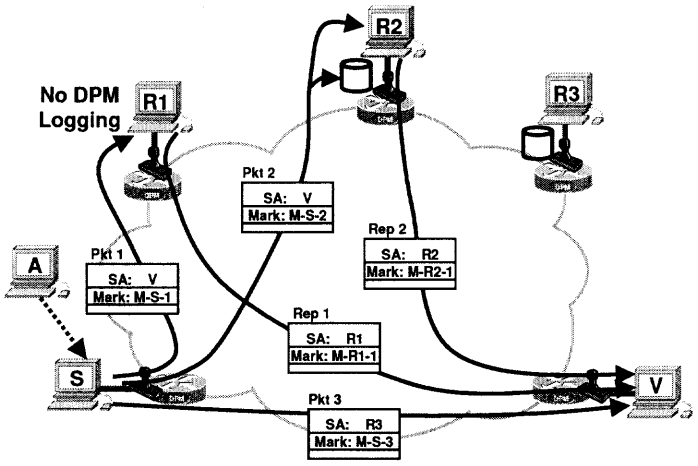

Figure 6.1 Composition of (D)DoS attacks.

In the most general case of DDoS attack, called a mixed DDoS attack in this dissertation, the attacker may instruct the slaves to flood the victim directly *and* send the traffic to the reflectors, from which packets are reflected to the victim. In Figure 6.1, slave **S** sends packets 1 and 2 to reflectors **R1** and **R2**, respectively. The SA of these packets are spoofed for the address of the victim **V**. The generated replies 1 and 2 are then directed to **V**. Notice that SA fields of the replies 1 and 2 are not spoofed and contain valid source addresses of **R1** and **R2**. Also **S** sends packet 3, with a spoofed SA for some random value **R3**, directly to **V**. The fact that **S** sends the packets to both **V** and **R**'s constitutes a mixed DDoS. A major attack would involve hundreds or even thousands of slaves and may involve up to a million of reflectors [37]. It is worth mentioning that the mixed DDoS attacks have neither been reported nor described in literature, and at this time remain purely theoretical.

The popular DDoS attacks, which are currently exercised, are the special cases of the mixed DDoS attack. One special case is what is known as a reflector-based

DDoS attack in which slaves send packets only to reflectors, and the victim is flooded by the replies from the reflectors only. The other special case is what is known as a slave-based DDoS attack, in which the slaves send packets only to the victim. The reflectors are not engaged in the slave-based attack. A special case of a slave-based DDoS attack is a DoS attack, in which only a single slave participates in the attack.

6.2 Traceback Data Structures

The DPM traceback module has to be integrated with the Intrusion Detection System (IDS) of the victim. The IDS will have to notify the DPM Traceback module on which incoming packets constitute the attack.

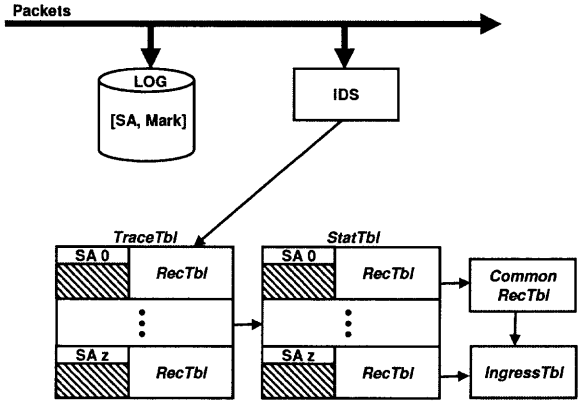

Figure 6.2 DPM traceback data structure.

As a stream of packets enters the victim's network, the SA field and the mark of *every* packet must be logged, even if it is not considered an attack packet. The logging must be done in order to be able to collect the marks of slaves' interfaces from the reflectors in the reflector-based or mixed DDoS attacks, as will be discussed in Section 6.4 and eventually reconstruct those ingress addresses, or to perform the traceback post-mortem. The IDS has to recognize the attack. The IDS can employ

signature-based detection, anomaly-based detection as defined by E. Carter in [36], and J. McHugh in [38], or a combination of both as described by T. Bass in [39]. The techniques of recognizing the attack are beyond the scope of this work. The end result of the IDS function is a collection of packets, constituting the attack. For every attack, which IDS recognizes, DPM Traceback Procedure will create an instance of a *TraceTbl*. The *TraceTbl* consists of a number of *RecTbl*s. Each *RecTbl* is associated with a source address of one of the attack packets. In addition, there is a *StatTbl*, a data structure identical to *TraceTbl*, that is associated with the *TraceTbl* and is used solely for analysis of the marks. Finally, there is a common *RecTbl* where final address reconstruction happens. The ingress addresses may be recovered in *StatTbl* or the common *RecTbl*, so it can be copied to *IngressTbl* from either data structure. Figure 6.2 illustrates the data structures involved.

6.3 Tracing Slaves from Reflectors

Having identified potential reflectors in the reflector-based or the mixed DDoS attack, it may be possible to determine the ingress addresses of the slaves. By examining the DPM logs on the reflectors, if they were kept, it is possible to extract the marks from the packets which caused the reflectors to send the attack packets to the victim, and to use those marks in the tracing procedure on the victim. Recall that the attacker engages a reflector in the attack by sending a packet to it from a slave with the SA spoofed for the address of the victim. The reply to this packet, whatever it might be, is directed to the victim. Even though the attacker may change the SA of the packets, the DPM marks cannot be changed. Therefore, the marks of the packets to

a reflector with the SA of the victim may be used to reconstruct the ingress address of the slave(s), which sent the packets to this reflector.

The protocol of obtaining the logs is beyond the scope of this work. A given reflector may have DPM logging enabled or disabled. When the victim makes a request to the reflector for the marks from the logs, the victim's address and the approximate time of packet arrival must be supplied. Three responses are possible:

- **Error (or No Response)**, if the logging is not enabled on a given reflector.
- **Positive Response**, with the list of marks matching the specified parameters returned to the requesting victim.
- **Negative Response**, if the logging on the reflector is enabled, but none of the logged entries matched the specified parameters.

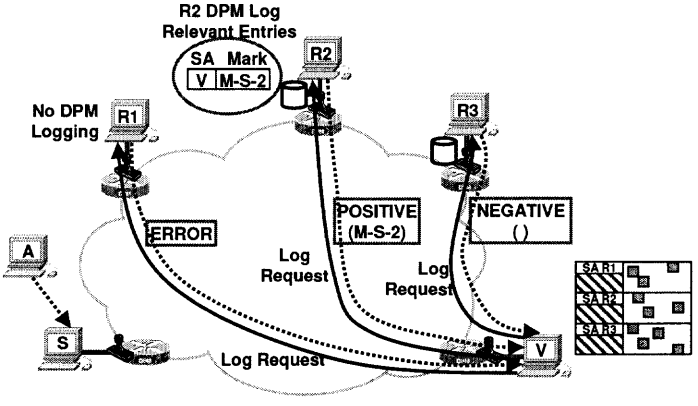

Figure 6.3 Illustration of reflector log requests and responses.

In Figure 6.3, three reflectors are shown. **R2** and **R3** have DPM logging enabled and **R1** does not. When **V** performs the traceback, the addresses of **R1**, **R2**, **R3** will be available to **V**. At this point, **V** has no knowledge that the packet

with SA of **R3** was sent from a slave and SA was spoofed. **V** will send log requests to each of these addresses. Figure 6.3 illustrates three possible responses to the log requests. When the log request is sent to **R1**, error (or no response at all – depending on implementation) is returned since the logging was not enabled. **R2** had logging enabled and had a record of packet with SA **V**. The marks (in this case only one) are sent in the response to **V**. **R3** also has logging enabled; however, it did not receive any packets with SA **V**, and so the response is negative. Both positive and negative responses are useful to the victim as will be seen in the next section.

6.4 Traceback Procedure

A single procedure must be able to handle all types of attacks discussed in Section 6.1. As the victim is being attacked, the attack packets will be identified. Every attack packet which arrives to the destination will have a mark. The appropriate bit of the *RecTbl* in the *TraceTbl* associated with the SA address of the attack packet will be set to '1' as described in Chapter 4.

Every T seconds or once the attack is over, the content of the *TraceTbl* is copied into *StatTbl*, which is used solely for the statistical analysis of the marks. The recording of the marks is not performed in *StatTbl*, although while the *StatTbl* is analyzed by the procedure, the incoming marks continue to be recorded in the *TraceTbl*. The *StatTbl* is used only to analyze the SAs and associated *RecTbl*s.

First, the traceback procedure makes log requests to potential reflectors. SAs which have *RecTbl*s associated with them would be used to address those reflectors. In case of a positive response from a reflector, the victim will obtain a list of marks

from the slaves. It is certain that those marks are from the ingress interface of one or more slaves, since they came in the packets that had the victim's SA, and so the marks received in the response are copied to the common *RecTbl*. It can be argued that the reflector could perform the traceback based on this marks by itself. However, the number of attack packets, which it would receive may not be enough for the IDS to recognize the attack, and even if the attack was recognized, the marks, which a single reflector would obtain, may not be enough for the traceback. In case of a negative response, the traceback procedure on the victim concludes that every mark in the *RecTbl* associated with that SA came from the ingress interface of one or more slaves. After the marks from this *RecTbl* are stored in the common *RecTbl*, this *RecTbl* is removed from the *StatTbl* and is not considered for further analysis. Only in the case of "no response", the procedure cannot make any conclusions and has to move on to the next SA in the *StatTbl*. These alternatives and the corresponding actions are depicted in Figure 6.4.

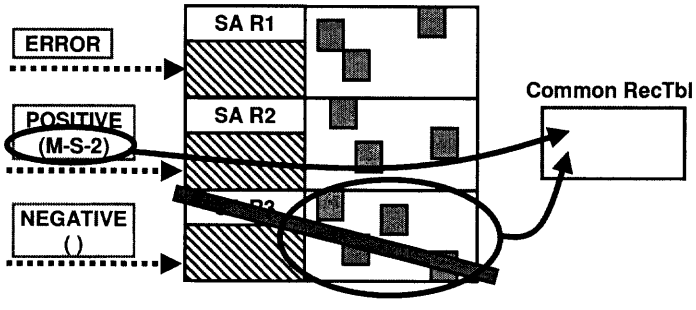

Figure 6.4 Processing of victim processing of log responses.

At this point, a *proper RecTbl* is defined as one where a single area of every part will have exactly k bits set to '1'. Moreover, none of the bits set to '1' can be within the same segment. If any of those conditions are violated, then a given *RecTbl*

is not proper. In other words, a proper *RecTbl* would have all marks necessary to reconstruct a single ingress address, and no other marks.

The procedure applies address recovery process to every individual *RecTbl*. Any ingress addresses, which are reconstructed, are stored in the *IngressTbl*. Moreover, if the *RecTbl* is proper, it is removed from the *StatTbl*. Refer to Sections 6.4.1 and 6.4.2 for further discussion on proper *RecTbl*s and mark deletion.

At this point, the procedure may have identified some marks from the ingress addresses of the slaves by analyzing the responses from the reflectors and copying them to the common *RecTbl*, and may have removed some reflector marks from the *StatTbl* by deleting the proper *RecTbl*s.

The number of potential marks from slaves' ingress addresses identified by the victim should not exceed $\overline{M}_{SL}$. $\overline{M}_{SL}$ is the expected number of marks, which the victim would collect if attacked by N_{MAX} attackers simultaneously. $\overline{M}_{SL}$ equals to the number of values a given segment in an area is expected to take, given by Eq. (4.2) multiplied by the number of segments in an area, multiplied by the number of areas in a part, and multiplied by the number of parts in a *RecTbl*.

$$\overline{M}_{SL} = f \times 2^d \times k \left[2^a - 2^a \left(1 - \frac{1}{2^a} \right)^{\frac{N}{E[H]}} \right],$$

where $E[H]$ is given by Eq. (4.1). For $f = 4$ and $k = 4$, $\overline{M}_{SL}$ is 32,038. This result was also supported by the simulation. Keeping the number of marks under $\overline{M}_{SL}$ will ensure that the rate of false positives will not exceed 1%.

Depending on the attack profile, some marks remaining in the *StatTbl* at this point may be from the slaves' ingress addresses. A certain number of these remaining marks should be selected to be copied to the common *RecTbl*. The total number of already copied marks and the marks to be selected from the ones still remaining in *StatTbl* should not exceed $\overline{M}_{SL}$. The mark occurrence is defined as the number of *RecTbl*s in which that mark appears. In other words, if a given mark arrived multiple times in the packets with the same SA, only one occurrence would be counted. The remaining marks with the highest number of occurrence would be selected. Assuming that the marks are distributed uniformly in the interval of $[0, 2^{17})$, the only reason for a certain marks to have a higher number of occurrence is that the slaves have sent packets to the victim with different SAs. This results in the situation when the same marks appear in the *RecTbl*s associated with different SAs, and thus its number of occurrence is increased. Therefore, marks with a higher number of occurrence are more likely to be from the slaves' ingress addresses. The marks with the highest number of occurrence are copied to the common *RecTbl*. The number of marks copied equals to $\overline{M}_{SL} - n(com.\,RecTbl)$, where $n(com.\,RecTbl)$ stands for the number of marks in the common *RecTbl*.

Finally, the address recovery process is applied on the common *RecTbl* and the ingress addresses are reconstructed. These can be the ingress addresses of slaves, reflectors which did not have a proper *RecTbl* associated with their SA, and false positives. A formal description of the procedure is presented in Figure 6.5.

```
Traceback procedure at victim V:
 for each SA in StatTbl
  send Log_Request(SA, V, time ± δ)
  if Log_Response ≠ NIL then
   read Marks[ ] of Log_Response
   record Marks[ ] ⇒ com. RecTbl
  if Log_Response == NIL then
   read Marks[ ] of SA.RecTbl
   record Marks[ ] ⇒ com. RecTbl
   delete SA.RecTbl from StatTbl
 for each SA in StatTbl
  run Address_Recovery(SA.RecTbl)
  if proper(SA.RecTbl) == TRUE then
   delete SA.RecTbl from StatTbl
 for each SA in StatTbl
  read Marks[ ] of SA.RecTbl
  for each Mark in Marks[]
   Occ_Structure[Mark].Value := Mark
   Occ_Structure[Mark].Occurrence ++
 Num_Select := M_SL − n(com. RecTbl)
 sort Occ_Structure[] by Occurrence
 for x := 0 to Num_Select
  record Occ_Structure[x].Value ⇒ com. RecTbl
 run Address_Recovery(com. RecTbl)
```

Figure 6.5 Pseudo code for the traceback procedure.

6.4.1 On Proper *RecTbl*s and Hiding of the Marks

In this section, a concept of a proper *RecTbl* is introduced. Recall that one of the steps of the procedure discussed above was to remove the proper *RecTbl* from the *StatTbl*. As part of the mixed DDoS, the attacker may attempt to send a packet from a slave with the SA of the reflector, which sent enough packets for the victim to collect a complete set of marks from its ingress interface. The attacker may try to create a situation where the mark in this packet would duplicate one of the marks

sent by the reflector. Thus, once it is established that this *RecTbl* is proper, this mark will be deleted.

Intuitively, the chance of this situation occurring is very low. The attacker may find few slaves' ingress addresses which have one or even two segments the same as a targeted reflector. The probability that at least one of the digests is the same multiplied by the probability that this digest and that segment are picked in the only mark sent with this SA make this situation highly improbable.

6.4.2 On Deleting Marks from *RecTbl*

If the victim undergoes a mixed attack, the attacker could instruct the slaves to send packets with the SA of the reflectors, thus making sure that the marks will be recorded in a *RecTbl* associated with some or all of the reflectors. By having additional marks in those *RecTbl*s, the attacker ensures that the marks from reflectors do not get deleted from the *StatTbl*. Why the marks which were used in reconstruction of a valid ingress address are not removed in the presence of other marks in the *RecTbl*? If the attacker came into possession of many slaves, it is possible to select reflectors in such a way that at least one out of $f \times k$ marks inserted by the DPM interface of the reflector is the same as one of the marks inserted by the DPM interface of the slave. The attacker may instruct the slave to send packets to the victim spoofing the SA to the SA of the reflector. If the slave does not send enough packets for the victim to collect all the marks in the *RecTbl* associated with the SA of the reflector so that the ingress address of that slave is reconstructed, then the ingress address of the slave will never be reconstructed if one of its marks is deleted. Therefore, the marks

may be deleted from the *StatTbl* only if the traceback procedure is certain with high probability that only the marks from a single reflector's DPM interface are recorded in the given *RecTbl*; in other words, the *RecTbl* is proper.

6.5 Conditions for Traceability and Untraceable Attacks

In this section, the attacks which can and cannot be traced back with the procedure described in Section 6.4 are analyzed. All of the attacks will be analyzed from the point of view of the victim traceback procedure. Denote S as the number of slaves involved in the attack, L as a factor of hosts on the Internet with enabled DPM logging, and C as the fragmentation coefficient.

Marginally traceable attacks are defined as the attacks during which the number of packets received by the victim, falls below the expected number of packets required for traceback. Yet, that number may be enough to collect all the marks necessary for traceback. *Untraceable* attacks are defined as the attacks which can never be traced. The difference between the two is that while *marginally untraceable* can still be traced with the probability of success of less than 50%, the *untraceable* attacks cannot be possibly traced. For the marginally traceable attacks with multiple hosts involved, such as DDoS attacks, the victim may be able to trace back to some of the hosts involved in the attack.

6.5.1 Intrusions

The intrusion, as mentioned in section 6.1.1, cannot have packets with the spoofed SA, and so theoretically a single packet identified by the IDS would be enough to perform the traceback. However, if the intruder engages in some elaborate scheme where his/her address is spoofed, but he/she is still capable of getting the desired information, DPM tracing procedure would have to be used. As mentioned before, artificial fragmentation will not provide any benefit to the attacker in terms of obstructing the traceback process, and therefore will not be utilized. The expected number of packets required for the traceback $E[Pkt]$ is then $E[D]$. Therefore, the marginally traceable intrusion with the spoofed SA, which is rare, would be the one which contains up to $E[D] - 1$ packets. The untraceable attack must consist of no more than $f \times k - 1$ packets.

6.5.2 DoS Attacks

The DoS attacks come from a single source, most likely with spoofed SAs in the attack packets, since the attacker is not interested in the replies from the victim. The number of marks required to be received would be $f \times k$ as described in Chapter 4. The artificial fragmentation could be used by the attacker to be able to send more packets before the traceback becomes possible. In order for the victim to be able to trace the ingress addresses of the slaves participating in a DoS attack, $C \times E[D]$ packets must be received as mentioned in Chapter 5. Sending less datagrams than $E[D]$ and $f \times k$ would produce marginally traceable and untraceable attacks, respectively. The respective number of packets would be $C \times (E[D] - 1)$ and $C \times (f \times k - 1)$.

6.5.3 Slave-based DDoS Attacks

The slave-based DDoS attack can be considered as a number of DoS attacks executed simultaneously. The victim for all of those attacks is the same. The expected number of packets required to be able to trace any of the slaves' ingress addresses is $C \times E[D]$, as discussed in Section 6.5.2. Therefore, the marginally traceable slave-based DDoS attack may consist of up to $S \times (C \times (E[D] - 1)$ packets. An untraceable attack must consist of no more than than $S \times (C \times (f \times k - 1))$ packets because in order for the whole attack to be untraceable, every slave must be untraceable.

6.5.4 Reflector-based DDoS Attacks

The reflector-based DDoS attack currently causes the most concern. The reflectors would be identified in the initial stage of the DPM tracing procedure. Identifying the reflectors is not a goal of the traceback. The reflectors are just the innocent servers with opened services used by the attacker to generate the traffic to the victim. The number of packets, which the collection of slaves may send in order to remain marginally traceable or untraceable, depends on the fraction of the hosts which perform DPM logging, and would be $\frac{S \times (E[D]-1)}{L}$ and $\frac{S \times (f \times k-1)}{L}$, respectively.

6.5.5 Mixed DDoS Attacks

The point that the attacks described in this section have not been reported or described yet, and are theoretical at this point is reemphasized. Yet, it is essential that the traceback scheme is capable of handling the unknown as well as known attacks.

If L is sufficiently large, then using reflectors becomes detrimental to the attacker's cause to have his slaves untraceable. The slave-based attack will allow the attacker to send more packets while remaining marginally traceable or untraceable. If, on the other hand, very few hosts on the Internet implement DPM logging, and L is small, then the reflector-based attack will allow the attacker to attack the victim with more packets while keeping slaves marginally traceable or untraceable. Notice that reflector-based DDoS and slave-based DDoS attacks are both special cases of mixed DDoS attacks. The number of packets which allows the attacker to wage marginally traceable attack is

$$\max\left\{\frac{S\times(E[D]-1)}{L},\ S\times C\times(E[D]-1)\right\},$$

and the number of packets for the largest untraceable attack is

$$\max\left\{\frac{S\times(f\times k-1)}{L},\ S\times C\times(f\times k-1)\right\}.$$

Two observations can be made. First, it observed that the number of reflectors is irrelevant for traceability of the attack. Notice that it does not appear in any of the expressions. Second, since $C=3$, about $1/3$ of hosts on the Internet must have DPM logging enabled so that it becomes detrimental, in terms of the number of packets required for traceability, to the attacker to engage reflectors. Table 6.1 summarizes the findings of this section.

Table 6.1 Maximum Number of Packets for Marginally Traceable and Untraceable Attacks

Type of the Attack	Marginally Traceable	Untraceable
Intrusion	$E[D]-1$	$f \times k-1$
DoS Attack	$C \times (E[D]-1)$	$C \times (f \times k-1)$
Slave-based DDoS	$S \times C \times (E[D]-1)$	$S \times C \times (f \times k-1)$
Reflector-based DDoS	$\frac{S \times (E[D]-1)}{L}$	$\frac{S \times (f \times k-1)}{L}$
Mixed DDoS	$\max\left(\frac{S \times (E[D]-1)}{L}, S \times C \times (E[D]-1)\right)$	$\max\left(\frac{S \times (f \times k-1)}{L}, S \times C \times (f \times k-1)\right)$

6.5.6 Storage Requirements

16KByte (2^{17} bits) of storage has to be allocated for every new SA involved in the attack. This number should be doubled since all *RecTbl*s are copied to the *StatTbl* for analysis. If millions of reflectors and slaves are involved in the attack, the storage requirements may be large. This may be an issue if the storage facilities are not properly sized. Currently the storage is a commodity and there are TByte hard drives available commercially [40]. The organizations interested in DPM should plan the storage capacity accordingly.

6.6 Simulation Results

Simulation results discussed in this section demonstrate the effectiveness of DPM Traceback. Several figures of merit are introduced to evaluate the performance of DPM which are affected by a number of independent variables. Several illustrative attack profiles, which cover major attack types were designed and simulated. The simulation results presented in this section should provide a good basis for the scheme evaluation.

6.6.1 Description of the Simulation

The following parameters were the inputs to the simulation:

- **Number of Reflectors.**
- **Packets Sent by Each Reflector.** The reflectors generate packets in response to the packets from the slaves. It is assumed that for every packet from a slave to a reflector, the reflector will generate a single packet.
- **Number of Slaves.** Even if the slaves do not attack the victim directly, such as in reflector-based DDoS attack, slaves generate packets to the reflectors. Slaves must participate in every type of attack.
- **Packets per Slave to the Victim.** Number of packets, sent by each slave directly to the victim. This does not include the number of packets, which each slave sends to reflectors.
- **Use of Artificial Fragmentation by the Slaves.** Artificial fragmentation allows a slave to send several packets with the same mark to the victim. All of these packets must have the same SA. So, on one hand, artificial fragmentation decreases the number of marks received from the slave for a given number of packets, which is undesirable, but, on the other hand, it decreases the number of potential reflectors from which the logs must be requested, which is desirable.
- **Percentage of Packets from the Slaves with Random SA.** Sending packets with the random SAs, not corresponding to the SAs of the reflectors has certain pros and cons from the attacker's point of view. Sending more packets

from the slaves with random SA will result in the increased number of negative responses from the potential reflectors. While this makes traceback more time consuming, it would have better chances of identifying the slave marks, and then ingress addresses of the slaves. Sending packets with SAs of reflectors will decrease the number of logs polled.

- **Percentage of Hosts with DPM Logging Enabled, L.** This parameter indicates the number of hosts which will produce a response to a log request.

A configuration of the above parameters is called an attack profile. The first six parameters would be controlled directly by the attacker. The last one, would not be controlled by the attacker, but is a major contributing factor in traceback effectiveness.

Some other data are directly related to the attack profile, yet are not explicitly controlled by the attacker:

- **Number of Traceable Slaves.** Traffic from every slave will pass through a DPM-enabled interface. If enough packets from a given slave pass through it, then a complete set of marks will reach reflectors and/or victim. A slave that sends enough packets to have its ingress interface traced, is called a traceable slave.

- **Total Number of Packets Sent to the Victim.** This signifies the severity of the attack. Usually, the more traffic is being sent to the victim the heavier is the impact of the attack.

- **Total Number of Marks from All Slaves' Ingress Interfaces.** This quantity is used by the simulation to establish how good the traceback is. In reality, the traceback procedure will never know how many marks were sent in the packets from the slaves.

According to the procedure described in 6.4, the first step is to send log requests. The simulation determines the following values:

- **Total Logs Requested.**
- **Number of Positive Responses.**
- **Number of Negative Responses.**
- **Number of errors (or No Responses).**

The respective percentages with respect to the total are also determined. After the victim has polled the logs, the following statistics are calculated:

- **Correctly Identified Marks from Slaves' Ingress Interfaces.** These are the marks identified, which indeed were inserted by the ingress DPM-enabled interfaces of the slaves. The percentage with respect to the total number of marks from all slaves is also calculated.
- **Not Identified Marks from Slaves' Ingress Interfaces.** These are marks, which are not identified by requesting logs from reflectors.
- **Incorrectly Identified Marks from Slaves' Ingress Interfaces.** There should be no incorrectly identified marks at this stage of the procedure.

However, when this statistic is recalculated later on, it indicates the number of marks, which were not inserted by slaves' ingress interfaces.

The number of correctly and not identified marks should equal to the number of marks from all slaves' ingress addresses. The incorrectly identified marks are a separate measure and refer to the amount of marks from reflectors' ingress addresses, which the traceback mistakes for the slaves' ingress addresses marks.

Finally, the marks with the highest occurrence are added to the common *RecTbl* as outlined in Section 6.4. (The simulation does not check if the *RecTbl*s in the *StatTbl* are proper. The output of the simulation, therefore, has a higher number of incorrectly identified marks from the slaves ingress interfaces as compared to what the actual procedure would have.) These steps are called additional processing at the victim. At this point, three final values are recalculated.

The simulation using the multiple digest DPM configuration with $f = 4$ and $k = 4$ is described in this section.

6.6.2 Description of Profiles

Seven representative profiles are presented here. The nature and voluminous amount of the data prevents us from discussing additional profiles.

6.6.2.1 Profile1. Profile 1 is a slave-based attack. 75% of hosts on the Internet support logging. Consequently, about 75% of the slave marks are identified by receiving negative responses to the log requests, which are sent to the spoofed addresses, since 100% of packets from slaves had a random SA. The total number of

logs requested is 166,666, which is approximately 1/3 of the total number of packets. This can be explained by the fact that artificial fragmentation was used, and the same mark was sent three times in the packets with the same SA. As expected, 75% of requested hosts produce negative responses, and the rest return no responses.

The remaining marks are copied to the *RecTbl* by the victim when the traceback procedure gets to the stage of selecting marks with the highest occurrence number. Since there were not enough marks in the *StatTbl* remaining to make the number of marks in common *RecTbl* to be $\overline{M}_{SL}$, all of the remaining marks were copied to the common *RecTbl*. 100% of slave marks were identified and copied to the common *RecTbl* by the end of the procedure, implying that all 1,000 slaves' ingress addresses will be reconstructed.

6.6.2.2 Profile2. Profile 2 is a reflector based attack. Notice that slaves do not send packets directly to the victim. In this scenario, 100% of hosts on the Internet have DPM logging enabled. This results in the full set of marks from slaves' ingress interfaces be recovered by polling the logs of the reflectors. Further analysis by the victim identifies 24,270 more marks to be copied to the *RecTbl*. Our simulation does not implement checking for the proper *RecTbl*. This checking would likely result in fewer incorrectly identified marks. When the reconstruction process is applied to the common *RecTbl*, all of the slave ingress addresses will be reconstructed along with some reflector ingress addresses and some false positives. Most importantly, all of the marks from slaves' ingress addresses are identified.

6.6.2.3 Profile3. Profile 3 is identical to Profile 2 except that there are no hosts on the Internet with DPM logging enabled. The outcome of the traceback procedure is significant. As expected, none of the slave marks were identified by requesting the logs since there was no response for any of the request. When the victim attempted to select the marks with the highest number of occurrence, only some marks were identified correctly. The reason for these correctly identified marks from slaves' ingress interfaces is that reflectors have the same marks and those 1,882 had a higher number of occurrence. In other words, these were the reflector marks, which coincided with the slave marks. The outcome of the traceback procedure on Profile 3 attack is consistent with the conditions for traceability outlined in Section 6.5.4, namely that reflector-based DDoS attacks cannot be traced if logging is not enabled on reflectors.

6.6.2.4 Profile4. Profile 4 is a mixed DDoS attack. 50% of hosts on the Internet have DPM logging enabled, and 50% of all packets from slaves have random SA. About 64% of all slave marks are identified by reflector log requests; the rest of them along with some other ones from the slaves are identified during the final processing by the victim.

6.6.2.5 Profile5. Profile 5 is similar to Profile 2 in that no hosts on the Internet implement DPM logging. However, the attack of this profile is a mixed DDoS attack, and the slaves send traffic to the victim as well. This allows the victim to identify all of the marks from the slaves after the request of reflectors' logs produced

no results. Also, notice that in this case, since slaves send a lot of packets to the victim, and all of them have random SA, more than 3.75M requests to the reflectors would have to be made.

6.6.2.6 Profile6. Profile 6 is a marginally traceable attack. Recall from Section 6.5 that in order to be marginally traceable a slave has to send between $f \times k$ and $E[D]$ datagrams during the attack. The slaves in this attack send 27 datagrams (3 fragments each datagram) each since artificial fragmentation is used for the traffic from the slaves to the victim. Most of the marks are identified by requesting the logs. The rest are identified during the final victim processing. Even though all of the marks are identified and copied to the common *RecTbl*, only 432 out of 1,000 slaves' ingress addresses will be reconstructed since only 432 sent enough packets to make a complete set of marks available.

6.6.2.7 Profile7. Profile 7 is an untraceable mixed DDoS attack. Recall from Section 6.5 that in order to be untraceable, a slave has to send less than $f \times k$ datagrams. Every slave sends 12 datagrams in total. Even though all of the marks from the slaves are identified and copied to the common *RecTbl* by the end of the traceback procedure, none of the slaves' ingress addresses will be reconstructed since none of them sent enough packets to have a complete set of marks for any of the ingress interfaces available to the victim and reflectors.

Table 6.2 Simulation Results for Selected Attack Profiles

	Profile	1		2		3		4		5		6		7	
Attack Characteristics Controlled Directly by the Attacker — Attack Profile	# of Reflector	0		10,000		10,000		10,000		3,000		1,000		1,000	
	Packets/Reflector	0		200		200		200		1,000		20		10	
	# of Slaves	1,000		500		500		1,000		2,250		1,000		1,500	
	Packets/Slave	500		0		0		500		5,000		80		20	
	Artificial Frag. used by Slaves	Yes		No		No		Yes		Yes		Yes		Yes	
	Packets from Slaves with random SA	100%		100%		100%		50%		100%		50%		0%	
	Hosts with DPM Logging Enabled	75%		100%		0%		50%		0%		100%		100%	
Attack Characteristics Indirectly Controlled or not Controlled by the Attacker	Traceable Slaves	1,000		500		500		1,000		2,250		432		0	
	Packets sent to Victim	500,000		2,000,000		2,000,000		2,500,000		14,250,000		100,000		40,000	
	Marks sent from Slaves' Ingress Interfaces	15,068		7,768		7,766		15,060		31,488		14,392		13,355	
Outcome of Requesting Reflector Logs	Total Logs requested	166,666		10,000		10,000		93,333		3,753,000		14,333		1,000	
	Positive responses	0	0%	10,000	100%	0	0%	5,000	5%	0	0%	1,000	7%	1,000	100%
	Negative responses	125,000	75%	0	0%	0	0%	41,667	45%	0	0%	13,333	93%	0	0%
	No responses	41,667	25%	0	0%	10,000	100%	46,667	50%	3,753,000	100%	0	0%	0	0%
After Reflector Log Requests, Marks from Slaves' Ingress Interfaces	Correctly Identified	11,492	76%	7,768	100%	0	0%	9,635	64%	0	0%	12,763	89%	8,127	61%
	Not Identified	3,576	24%	0	0%	7,766	100%	5,425	36%	31,488	100%	1,629	11%	5,228	39%
	Incorrectly Identified	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%
After Additional Processing at Victim, Marks from Slaves' Ingress Interfaces	Correctly Identified	15,068	100%	7,768	100%	1,882	24%	15,057	100%	31,488	100%	14,392	100%	13,355	100%
	Not Identified	0	0%	0	0%	5,884	76%	3	0%	0	0%	0	0%	0	0%
	Incorrectly Identified	0	0%	24,270	76%	30,156	94%	16,981	53%	550	2%	9,887	41%	6,638	33%
Comments		Slave-based DDoS		Reflector-based DDoS		Reflector-based DDoS – no logging		Mixed DDoS		Mixed DDoS – no logging		Marginally Traceable Mixed DDoS		Un-traceable Mixed DDoS	

CHAPTER 7

DPM DEPLOYMENT AND TOPOLOGICAL ISSUES

Up until now the Internet was considered to be a homogeneous network with a single administration and perfect DPM deployment on the edges of the network. In reality, the Internet is not, and has never been, a single homogeneous network according to G. Huston [41], [42], and J. Rexford [43]. The Internet, from its very beginning, was an interconnection of different networks. Over time the Internet became more structured, but its heterogeneous nature still remains. In this section the structure of the Internet is described and the relationships among the Autonomous Systems (ASs) are explored. Then, the simple guidelines for deploying DPM are introduced and analyzed.

7.1 Structure of the Internet

The Internet is a hierarchical structure [41], [42]. Several ISPs, so called tier 1 ISPs, constitute the backbone of the Internet. Very few ISPs, such as UUNET, SprintLink, AboveNet, GBLX, AT&T and a few others, have the status of tier 1. The recent visualization of the Internet, where the hierarchy can be observed, is available on the web site of CAIDA [44]. These ISPs have a large geographical presence to facilitate a convenient connection of other ISPs. The ISPs of the second tier do not have the geographical presence comparable to tier 1 ISPs. Therefore, in order to establish connectivity to the other parts of the world, tier 2 ISPs have to buy transit services from one or more tier 1 ISPs. The next tier of ISPs have even less geographical presence. Yet, these ISPs are not at the lowest levels of hierarchy. These ISPs have to

buy transit services from the upper tier ISPs in order to establish global connectivity. These ISPs of medium tiers are often called regional ISPs. Finally, there are lowest tier ISPs, which sell transit services only to the retail customers such as home users and businesses. It is also suggested by A. Feldmann and J. Rexford [45] that the ISP's IP networks themselves have a hierarchical structure with well defined *core* links and *edge* links.

An administrative domain, such as an ISP or an enterprise network, may consist of more than one AS according to L. Gao [46]. For example, when two organizations, each with its own AS, merge, the resulting network will be a single administrative domain encompassing two ASs. The connectivity among the ASs is provided by the Border Gateway Protocol (BGP), version 4 [47]. BPG is used for the exchange of routes and for the selection of routes based on the predefined policies. Using BGP an AS advertises certain routes to its neighbors. The commercial relationships between the ASs define the rules, also called *policies*, of route advertising. In the rest of this section, relationships between ASs classified by L. Gao [48, 46], which do not always fit the purely hierarchical structure will be examined. Figure 7.1 illustrates all of the relationships described later in this section.

7.1.1 Customer-to-Provider Relationship

In the customer-provider relationship, the provider, usually an AS of a higher tier ISP, advertises all known routes to the customer, an AS of the lower tier ISP. The customer pays for this *transit service* and advertises all of its known routes to the

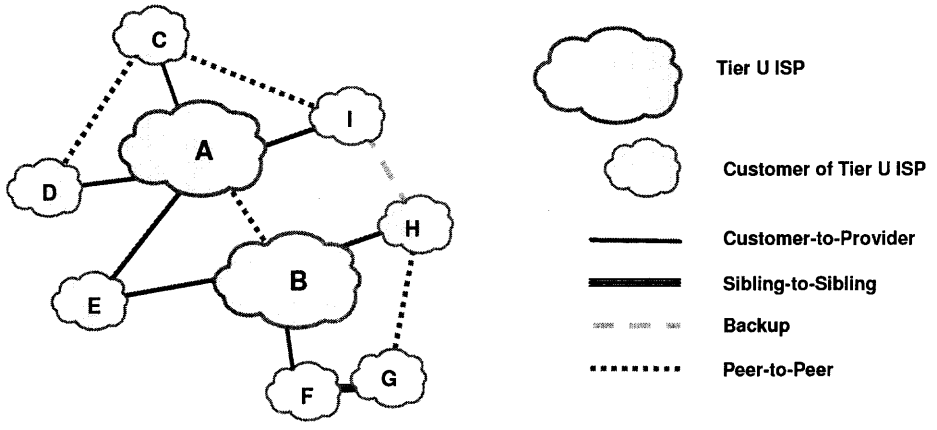

Figure 7.1 Illustrations of inter AS relationships.

provider. The provider in one relationship may be a customer in another relationship, and the customer in one relationship may be a provider in another relationship.

For increased availability, a given customer may enter a customer-provider relationship with more than one provider. Such an arrangement is called *multi-homing.* The end customer, such as a local ISP or an enterprise, that maintains customer-provider relationship with a single provider is called a *stub.* In a multi-homed arrangement, the customer does not advertise the providers' routes. Otherwise, the customer's network may be used as a transit between its providers.

It should also be noted that the customers do not necessarily have to connect only to the local, lowest tier, ISPs only. ISPs of high tiers offer retail services as well, and the end customers may purchase transit services from them directly.

In Figure 7.1, the ASs **C**, **D**, **E**, and **I** have the customer-to-provider relationship with AS **A**, and ASs **E**, **H**, and **F** have the customer-to-provider relationship with AS **B**. Autonomous system **E** has the customer-to-provider relationship with both ASs **A** and **B**, and therefore, AS **E** is multi-homed.

7.1.2 Peer-to-Peer Relationship

Two ISPs may find it mutually beneficial to exchange the routes to its customers directly. This is called a peering arrangement and is usually free of charge to both parties since it usually provides equal benefits to both ISPs. In order to establish the peering arrangement, the respective ASs have to enter a peer-to-peer relationship. In the peer-to-peer relationship the routes to the ASs' own customers are usually advertised to the peer AS. The peering arrangements are not associative, meaning that two peers of a given AS are not peers of each other, unless they setup an explicit peer-to-peer relationship between each other.

Peer-to-peer relationships are established usually for economic reasons. Instead of sending traffic through the transit network, it is more efficient and cost effective to send the traffic directly to the peer. This reduces the amount of traffic which is sent to the provider, as well as offers better service between the customers of the two peers.

There are several instances of peer-to-peer relationship in Figure 7.1. Autonomous system **A** and **B** have a peer-to-peer relationship. Therefore, the customers of **A** would be able to communicate with the customers of **B** without using the providers of **A** and **B** (not shown in Figure 7.1). Autonomous system **C** has a peer-to-peer relationships established with **D**, and with **I**. It means that the traffic from customers of **C** will be able to reach the customers of **D** and **I** without traversing **A**. However, the traffic from customers of AS **I** will have to traverse AS **A** in order to reach customers of AS **D**, and vice-versa. There is also a peer-to-peer relationship between ASs **G** and **H**.

7.1.3 Sibling-to-sibling Relationship

In the sibling-to-sibling relationship, the two ASs exchange the routes to its customers, peers and providers. In other words, the two sibling ASs share all routes. This arrangement is usually used for the ASs, which belong to the single administrative domain. Also, when two stubs cannot afford a transit service by themselves, they may effectively combine their networks by entering the sibling-to-sibling relationship and use a single connection to the provider, provided the bandwidth requirements are met, in order to save money.

Autonomous systems **F** and **G** in Figure 7.1 have a sibling-to-sibling relationship. Since they exchange all of their available routes, customers of **G** will have access to the rest of the Internet. Similarly, although **F** does not have an explicit peer-to-peer relationship with **H**, the peer-to-peer relationship between them will exist implicitly since **G** would share routes to the **F**'s customers with **H**, and **H**'s routes with **F**.

7.1.4 Backup Relationship

Two ASs may have a backup relationship, given that they have different providers. If the provider of one of the AS in the backup relationship fails, then this AS will use its backup AS as a transit network to the Internet. The purpose of this arrangement is similar to that of the multi-homing described in Section 7.1.1. However, instead of actually purchasing transit service from two providers, the customer purchases the transit service from one provider and enters a mutual backup relationship with another AS, which is cheaper.

Autonomous systems **H** and **I** in Figure 7.1 have a backup relationship. This means that if for some reasons, AS **B** would become unavailable to **H**, it would be able to maintain the connectivity to the rest of the Internet through ASs **I** and **A**. Similarly, if AS **I** would loose its connectivity to **A**, it would still maintain the connectivity to the rest of the Internet through ASs **H** and **B**. The customers do not necessarily have to connect only to the local ISPs only. In reality, as seen in Figure 7.2, the customers may connect directly to the ISPs of the high tiers, where **B** has two retail customers.

7.2 Ideal DPM Deployment

The ideal DPM deployment is a situation, where every interface connecting to the customers of all ISPs in the Internet would be DPM enabled. This situation is depicted in Figure 7.2.

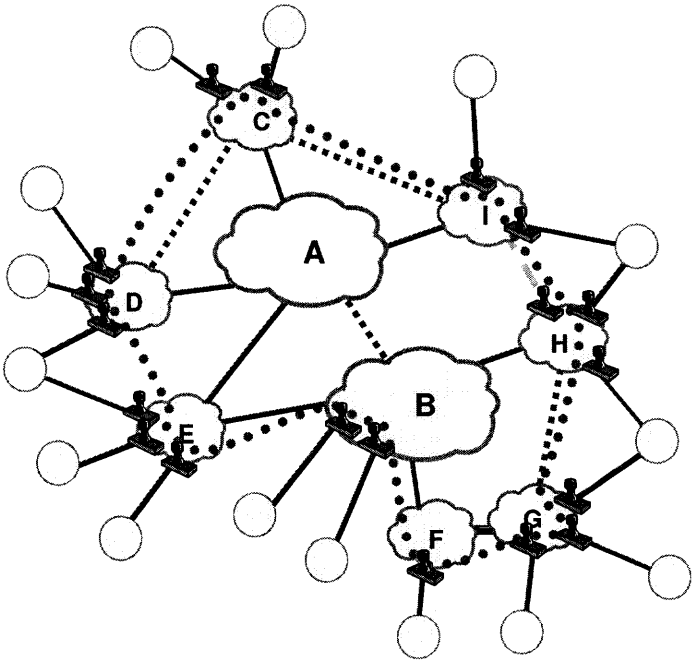

Figure 7.2 Ideal DPM deployment.

The collection of DPM-enabled interfaces is called a *DPM perimeter*. The perimeter must not have any *holes*, the access points through which the traffic from a customer may traverse the Internet unmarked. The DPM perimeter is shown in Figure 7.2 as a thick dotted line with circular dots.

7.3 Guidelines for DPM Deployment

It cannot be expected that all ISPs will deploy DPM simultaneously, if at all. Also even if that is to happen, DPM has to be disabled on some interfaces, as it is being enabled on the others to maintain the DPM perimeter. Therefore some coordinated effort on behalf of the ISPs is needed.

Neighboring ASs are defined as two ASs, which have at least one external BGP (eBGP) session setup between them. It is possible to have more than one eBGP session between a pair of ASs. Also, it is necessary to make a natural assumption that an ISP is not acting maliciously. In other words, if the ISP claims to have deployed DPM, then it can be trusted that DPM is in fact deployed and it is deployed according to the guidelines described below.

Tier 1 ISPs must have DPM enabled on all edge interfaces of all their ASs, except those which face other ASs of tier 1 ISPs. This can be accomplished by the good will of these ISPs or by forcing them to do so. All of the tier 1 ISPs are US-based companies, thus making application of some legal responsibilities possible.

After the Internet's core has been DPM enabled, the rest of the ISPs can join the DPM scheme gradually. Only the ASs, whose providers have deployed DPM, may enable DPM on its edges. Once the AS of the lower tier enabled DPM on its edges,

this fact has to be communicated to the provider. The provider must then disable DPM on the interface to this customer.

If there are two neighboring ASs, which are in either sibling-to-sibling, peer-to-peer, or backup relationship and both of them have DPM enabled, then both should disable DPM on the interfaces facing each other. This requires that ISPs share the information about DPM deployment, but it is assumed that administrations of ASs, which established any relationship would have means and incentives to communicate this information to each other.

7.3.1 Illustrative Example

Consider the network introduced in Figure 7.1 discussed so far. For illustrative purposes, it is assumed that ASs **A** and **B** are tier 1 ISPs and the rest of the ASs are tier 2. As discussed earlier, **A** and **B** have to deploy DPM before the rest of the ASs, then the following sequence of AS deployment is considered: **F**, **H**, **C**, **E**, **D**, **I**, and **G**.

Autonomous systems belonging to the tier 1 ISPs must deploy DPM first on their edge interfaces. Since AS **A** and AS **B** have a peer-to-peer relationship and both of them have DPM enabled, the DPM must be disabled on the interfaces between them, as can be seen in Figure 7.3.

Next, DPM is deployed on the edge interfaces of the first tier 2 ISP **F** as seen in Figure 7.4. Autonomous system **F** has a single interface to **B** and since both of them will have DPM enabled, **B** has to disable DPM on its interface to **F**. If **B** does not disable DPM on the interface to **F**, then the attack traffic from the **F**'s customers

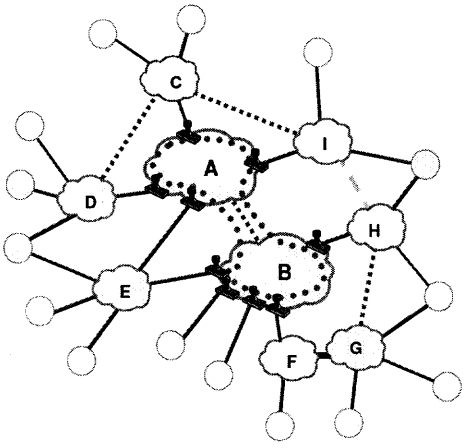

Figure 7.3 DPM deployment example, DPM enabled on tier 1 ISPs **A** and **B**.

will be marked by the edge interface of **F**, and then those marks will be over-written by the marks of **B**. So, effectively the traceback will be possible only to the edge of **B**. Autonomous system **F** has a sibling-to-sibling relationship with AS **G**. Since **G**, at this moment had not deployed DPM on its edges, **F** has to deploy DPM on the interface to **G**.

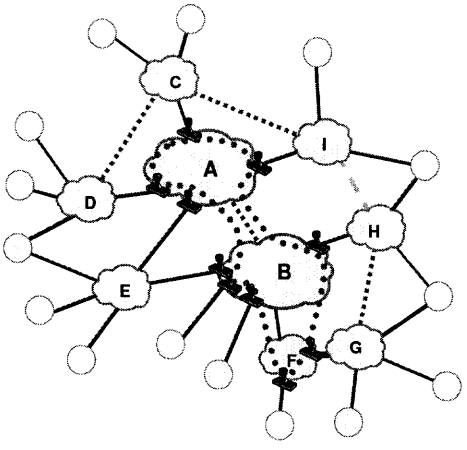

Figure 7.4 DPM deployment example, DPM enabled on AS **F**.

Next, AS **H** deploys DPM on its edges as shown in Figure 7.5. Since ASs **I** and **G**, which have sibling-to-sibling and peer-to-peer relationships with H respectively, do not support DPM yet, **H** has to deploy DPM on its edge interfaces to these ASs. Autonomous System **B** has to disable DPM on its interface to **H** in order not to over-write the marks by **H**'s edge interfaces.

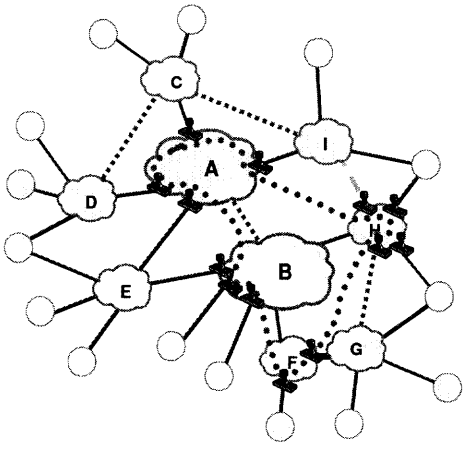

Figure 7.5 DPM deployment example, DPM enabled on AS **H**.

Next, AS **C** deploys DPM on its edge interfaces and **A** disables DPM on the interface to **C** as shown in Figure 7.6. Since peers of **C**, **I** and **D** do not have DPM deployed yet, **C** has to enable DPM on all of its edges, except for the interfaces to AS **A**. **A** also has to disable DPM on the interface to **C**.

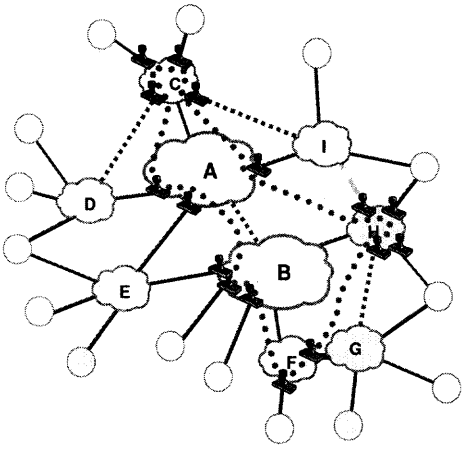

Figure 7.6 DPM deployment example, DPM enabled on AS **C**.

Autonomous System **E** is multi-homed; it is a customer of both AS **A** and AS **B**. This does not complicate the DPM deployment process. Since **E** does not maintain any relationships with any other ASs other than **A** and **B**, DPM has to be enabled on all of its edge interfaces, except the ones facing **A** and **B**. **A** and **B** have to disable DPM on the interfaces to **E** as shown in Figure 7.7

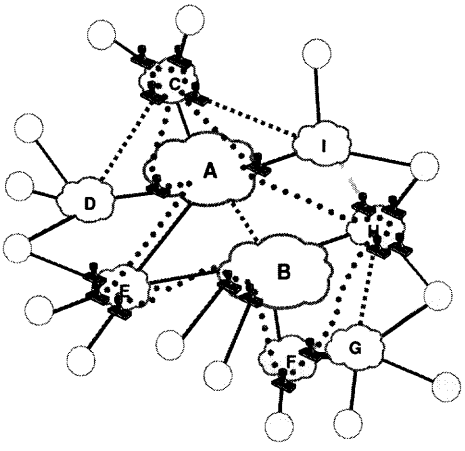

Figure 7.7 DPM deployment example, DPM enabled on AS **E**.

When AS **D** enables DPM on its edges, not only AS **A**, but also its peer, AS **C**, has to disable DPM on the interfaces to **D** because at that point both **C** and **D** will have DPM deployed as shown in Figure 7.8.

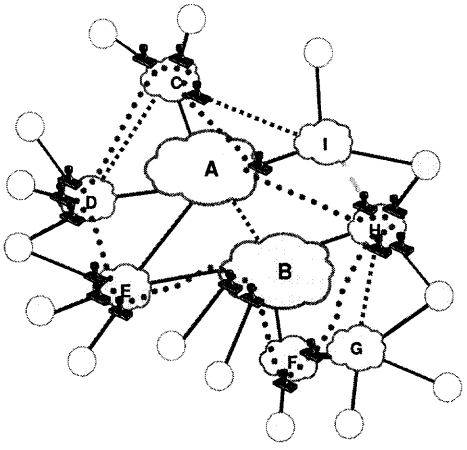

Figure 7.8 DPM deployment example, DPM enabled on AS **D**.

Similarly, when AS **I** enables DPM on its edge interfaces, the interfaces to **C** and **H** would not need to have DPM enabled on them since the respective ASs have DPM enabled. Also the DPM would have to be disabled on **C**'s and **H**'s interfaces to **I**. The resulting DPM perimeter is shown in Figure 7.9.

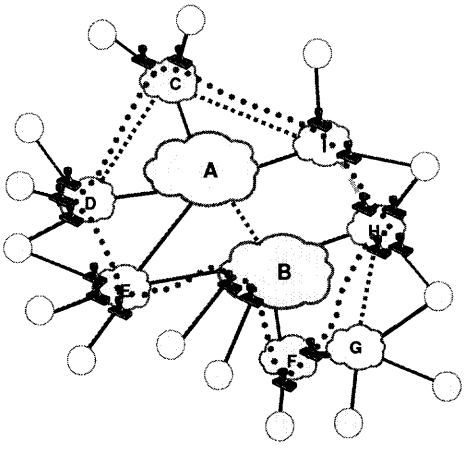

Figure 7.9 DPM deployment example, DPM enabled on AS **I**.

Finally, when DPM is enabled on the edges of AS **G**, it has to be disabled on the interfaces of its peer **H** and of its sibling **F**, which provides connectivity to **G** as shown in Figure 7.10.

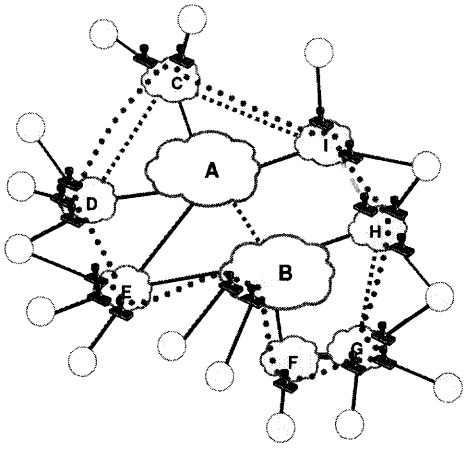

Figure 7.10 DPM deployment example, DPM enabled on AS **G**.

If it is assumed for illustrative purposes that the network described so far is a complete Internet, then ideal DPM deployment will result after AS **G** deploys DPM on its edges.

CHAPTER 8

DISCUSSION AND CONCLUSIONS

Traditionally, all IP traceback schemes perform what is known as the full path traceback, where a complete path of the attack packets through the Internet is determined. Deterministic Packet Marking does not perform full path traceback. Only the closest to the source interface which belongs to the DPM perimeter is determined during the traceback. In this chapter, the advantages and the disadvantages of the DPM traceback versus the full path traceback will be addressed.

It can also be argued that the ingress address filtering, described by P. Ferguson [5], is as effective as DPM if deployed around the same perimeter. This argument is addressed in this chapter as well.

Finally, some concluding remarks on DPM and traceback in general are made.

8.1 Comparison of DPM to Full Path Traceback Schemes

Deterministic Packet Marking can be viewed as a special case of PPM described in Section 2.2.1 and introduced by Savage, et al. [6]. The differences are that marking happens only at the edges of the collection of the deployed networks and the probability of marking is 100%. So what is lost and what is gained by these changes?

Deterministic Packet Marking must be deployed according to the guidelines outlined in Chapter 7. This requires the synchronization of efforts on behalf of the ISPs. Probabilistic Packet Marking, on the other hand, can be deployed independently on every ISP. Consider the situation depicted in Figure 8.1, where ISP **O** of tier U does not deploy a traceback scheme, and ISPs **P** and **R** of tier $U+1$ do deploy

the traceback scheme. Also, the attack path from the attacker **A** to the victim **V** traverses the path **P-O-R**.

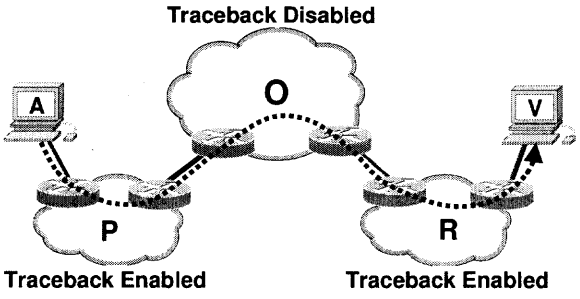

Figure 8.1 Example of partial traceback deployment.

If a full-path traceback scheme is deployed, the partial path encompassing the routers in **P** and **R** will be reconstructed. If DPM is deployed, the address of the **R**'s interface to **O** will be reconstructed. The marks of the ingress interface of **P** inserted in the packets will be overwritten by the marks of the **R**'s interface to **O**. That is exactly why this situation is not possible in DPM, if the deployment guidelines described in Chapter 7 are followed.

Deterministic Packet Marking cannot trace the attacks which were initiated from inside of the DPM perimeter, situation depicted in Figure 8.2. There is an unsubstantiated opinion that the attackers subvert one or more routers as a part of most attacks. In reality, subverting a router is a difficult task, usually possible as a result of an improper router configuration. To get a feel for how vulnerable is the network equipment compared to the workstations, the Computer Emergency Response Team (CERT) vulnerability notes database [49] was examined. A vulnerability is a flow in the system that can be used to take full or partial control over the system, or just bring it down. Vulnerability notes database is a collection of known vulnerabilities,

which have been reported so far. At the time of this writing, May 12th, 2003, there were 825 known vulnerabilities in the database. Only 31 (under 4%) of them were attributed to the ISP grade network equipment, the rest to various software packages of different platforms, and some to home office or Local Area Network (LAN) network equipment. Of these 31, only a handful were severe enough to allow an attacker to take control over the device. The low percentage of ISP grade network equipment vulnerabilities may serve as an indication of how difficult it is to accomplish the situation depicted in Figure 8.2. It can also be concluded that in the vast majority of the attacks, the attack packets are generated by the workstations, and therefore are traceable by DPM.

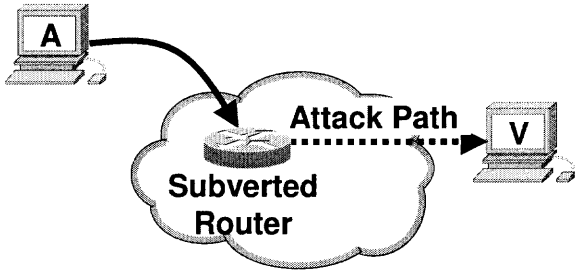

Figure 8.2 Situation of the subverted router inside an ISP.

There are many advantages of DPM over the full-path marking schemes. Security issues of PPM-like schemes arise from the fact that an attacker can inject a packet, which is marked with erroneous information. Such behavior is called *mark spoofing*. Prevention of such behavior is accomplished by special coding techniques, and is not 100% proof. Deterministic packet marking ensures that every packet, which arrives to the victim is correctly marked, and thus the need in those complex and processor intensive encoding techniques is unnecessary.

The following observation about all full-path traceback schemes is made: in a datagram packet network, the full-path traceback is as good as the address of an ingress point in terms of identifying the attacker. By definition, each packet in a datagram network is individually routed. Since every packet may take a different path from the source to the destination, only the ingress interface on the router closest to the source must be the same. Packets may take different routes even if their source and destination are identical. This may happen for two reasons: due to unwanted isolations of the network routing, or due to desired bandwidth management techniques such as load balancing. The changes in route between the source of the attack packets and the victim will be detrimental for the full-path traceback since more than one path for the single source would be reconstructed. This, however, does not affect DPM.

Internet service providers may only use public addresses for interfaces to customers and other networks, and use private addressing plans within their own networks. In this case, the usefulness of the full-path traceback becomes very low since information produced for the most part cannot tell the victim much more than a few IP addresses on the borders between ISPs. Even, if this is not the case, and public addressing is used within ISPs' networks, ISPs generally feel reluctant to disclose their topologies. Full path traceback schemes reveal topology of all networks by design. To limit this undesirable behavior, only routers, whose addresses are already known, should implement such schemes.

8.2 Comparison of DPM to Ingress Address Filtering

Source address filtering is a mechanism of ensuring that only the packets with the valid SAs are entering the Internet. The range of valid addresses is setup manually on the ingress interface and usually corresponds to the range of IP addresses of the hosts, which are expected to connect to the Internet through this interface.

Ingress address filtering is usually associated with high processing overhead. It is usually a subset of a large filtering mechanism, which enables filtering of packets by many other fields of layer 3 and 4 headers in the packets. To perform this filtering, every packet has to be taken off the fast switching hardware-based path and be analyzed by software, thus drastically increasing the processing overhead incurred by the router for every packet. The processing overhead, however, is not what precludes ingress address filtering from becoming an effective protection against the DDoS attacks. Hardware based mechanism, which would filter packets based on their SA only, would not be difficult to implement. In fact, the processing overhead incurred for every packet on the routers would be comparable to DPM, and none of the victim's processing would be necessary.

Unfortunately, ingress filtering is only effective in preventing DDoS attacks unless carried out everywhere according to the article of K. Park and H. Lee [50]. In addition, the ingress filtering has to be constantly managed if it is not deployed on the edges of the network, and its effectiveness is severely degraded. Consider Figure 8.3 where ISP **O** of tier U does implements ingress address filtering, and ISPs **P** and **R** of tier $U+1$ do not. The customer network connects to ISP **P**. Assume also that at some point in time ISP **O** is aware of the address ranges used by both

lower tier ISPs and the customer. If the customer changes its provider from **P** to **R**, then the ranges on the interface performing the ingress filtering on the edges of **O** have to be reconfigured to allow the traffic from the customer through **O**. In case the customer changes its IP address range, similar reconfiguration tasks would have to be performed. If the customer is multihomed, then **O** would have to accept traffic from customers with SAs of the interfaces from **R** and **P**. In other words, any workstation could spoof the source address of their packets to the ones of this customer, and this traffic would be allowed to pass.

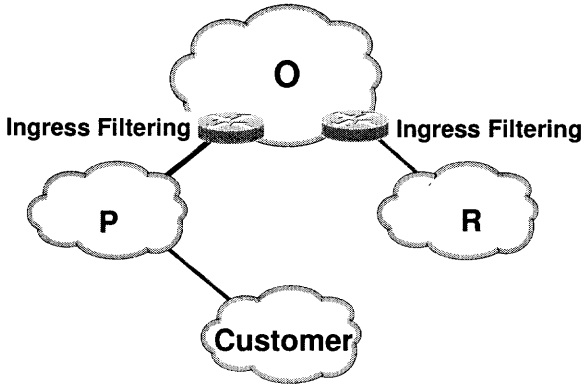

Figure 8.3 Limitations of ingress address filtering.

Once the attacker finds a way for his or her packets with the spoofed SAs to pass through the ingress filtering, the attack cannot be stopped. DPM, however, even if deployed on the same networks as the ingress address filtering, would provide the victim with concrete IP address, where the attack traffic entered the DPM perimeter.

8.3 IP Traceback Implications and Challenges

In addition to the technical aspects of IP traceback, there are also legal and societal aspects [51]. Several US federal laws that are relevant to traceback were not written with computer networking in mind. Currently, an insufficient number of court cases and precedents make it difficult to understand all implications of the traceback.

Collecting packet headers is generally not considered intrusion of privacy, and is legal. Collecting payloads, on the other hand, is illegal. Collecting digests is currently a gray area. Schemes like PPM, Overlay Network, Traceback with IPSec, or Controlled Flooding discussed in Sections 2.2.1, 2.2.3, 2.2.6, 2.2.5, respectively, do not collect any data from the payload, and the results of traceback may be admissible in court. On the other hand, ICMP Traceback and Hash-based IP Traceback schemes discussed in Sections 2.2.2 and 2.2.4 collect either digest or actual content of the packet, and may not be admissible. The developers of IP Traceback schemes have to be aware of legal implications, and that these methods can potentially intrude into privacy of individuals and corporations. According to [51], privacy of information is a higher priority than attack traceback even for organizations which may become a target of the attack, and the incentive for implementing traceback schemes is minimal.

Policy implications are also very important. Legislation, which requires IP Traceback, may be needed for ISPs to start implementing and deploying the schemes. This is a big problem in itself. Resolving this problem may not be enough since other countries do not have to comply with US laws. In case of non-compliance, any traceback solution would be able to conduct traceback as far as to the edge of the

compliant network. Non-compliant countries may become a safe-haven for attackers. The attacks may initiate from them or go through them and will be untraceable.

The issues discussed here are only the tip of the iceberg of many legal, political and societal issues which traceback may involve. The developers of the schemes should keep in mind that even the best traceback solutions from a technical stand point may be unsuitable for implementation and deployment for non-technical reasons.

8.4 Conclusions

Coping with computer attacks is a complex, multifaceted problem. There are different philosophies in the academia and the industry on how to deal with them. The combination of different methods in prevention, and traceback is what most likely be ultimately used in dealing with computer crimes. In this dissertation, Deterministic Packet Marking – an approach to IP traceback which effectively combines packet marking mechanism with the ingress only processing – was presented. It has better performance characteristics than other marking schemes. Most importantly, it was demonstrated that it is possible to trace the slaves in the DDoS attacks involving reflectors by DPM. This the first effort to a comprehensive traceback, which is capable to trace beyond the reflectors.

REFERENCES

1. S. E. Cross, "Cyber security," Testimony before the Senate Armed Services Committee, Mar. 2000.

2. R. D. Pethia, "Computer security," Testimony before the Committee on Government Reform, Mar. 2000.

3. S. Gibson, "DRDoS: Distributed reflector denial of service," Gibson Research Corporation," Technical Report, Feb. 2002.

4. S. Gibson, "The strange tale of the denial of service attacks against grc.com," Gibson Research Corporation," Technical Report, Mar. 2002.

5. P. Ferguson and D. Senie, "Network ingress filtering: defeating denial of service attacks which employ IP aource address spoofing," RFC 2827, May 2000.

6. S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Network support for IP traceback," *IEEE/ACM Trans. Networking*, vol. 9, no. 3, pp. 226–237, June 2001.

7. C. Morrow and B. Gemberling. How to track a DoS attack (NANOG post). [Online]. Available: http://www.secsup.org/Tracking/

8. D. Moore, G. M. Voelker, and S. Savage, "Inferring internet denial of service activity," in *Proc. of 10th USENIX Security Symposium*, 2001, pp. 9–22.

9. R. K. C. Chang, "Defending against flooding-based distributed denial-of-service attacks: A tutorial," *IEEE Commun. Mag.*, vol. 40, no. 10, pp. 42–51, Oct. 2002.

10. A. Belenky and N. Ansari, "On IP traceback," *IEEE Commun. Mag.*, vol. 41, no. 7, pp. 142–153, July 2003.

11. P. Srisuresh and K. Egevang, "Traditional IP network address translator (traditional NAT)," RFC 3022, Jan. 2001.

12. D. X. Song and A. Perrig, "Advanced and authenticated marking schemes for IP traceback," in *Proc. of INFOCOM 2001*, vol. 2, Apr. 2001, pp. 878–886.

13. D. Dean, M. Franklin, and A. Stubblefield, "An algebraic approach to IP traceback," *ACM Trans. on Information and System Security (TISSEC)*, vol. 5, no. 2, pp. 119–137, May 2002.

14. T. W. Doeppner, P. N. Klein, and A. Koyfman, "Using router stamping to identify the source of IP packets," in *Proc. of 7th ACM Inter. Conf. on Computer Comm. and Networks*, Nov. 2000, pp. 184–189.

15. S. M. Bellovin, "ICMP traceback messages," IETF Draft, Mar. 2000. [Online]. Available: http://www.research.att.com/ smb/papers/draft-bellovin-itrace-00.txt

16. S. F. Wu, L. Zhang, D. Massey, and A. Mankin, "Intention-driven ICMP trace-back," IETF Draft, Feb. 2001. [Online]. Available: http://www.silicondefense.com/research/itrex/archive/tracing-papers/draft-wu-itrace-intention-00.txt

17. S. F. Wu, L. Zhang, D. Massey, and A. Mankin, "On design and evaluation of 'intention-driven' ICMP traceback," in *Proc. of 10th Inter. Conf. on Computer Comm. and Networks*, Oct. 2001, pp. 159–165.

18. R. Stone, "CenterTrack: An IP overlay network for tracking DoS floods," in *Proc. of 9th USENIX Security Symposium*, Aug. 2000.

19. A. C. Snoeren *et al.*, "Single-packet IP traceback," *IEEE/ACM Trans. Networking*, vol. 10, no. 6, pp. 721–734, Dec. 2002.

20. S. Matsuda, T. Baba, A. Hayakawa, and T. Nakamura, "Design and implementation of unauthorized access tracing system," in *Proc. of the 2002 Symposium on Applications and the Internet, 2002. (SAINT 2002)*, Jan./Feb. 2002, pp. 74–81.

21. T. Baba and S. Matsuda, "Tracing network attacks to their sources," *IEEE Internet Comput.*, vol. 6, no. 2, pp. 20–26, Mar./Apr. 2002.

22. H. Burch and B. Cheswick, "Tracing anonymous packets to their approximate source," in *Proc. of 2000 USENIX LISA Conference*, Dec. 2000, pp. 319–327.

23. H. Chang *et al.*, "DECIDUOUS: Decentralized source identification for network-based intrusions," in *Proc. of 6th IFIP/IEEE International Symposium on Integrated Net. Management*, May 1999, pp. 701–714.

24. H. Chang *et al.*, "Design and implementation of a real-time decentralized source identification system for untrusted ip packets," in *Proc. of the DARPA Information Survivability Conference & Exposition*, vol. 2, Jan. 2000, pp. 100–111.

25. A. Belenky and N. Ansari, "IP traceback with deterministic packet marking," *IEEE Commun. Lett.*, vol. 7, no. 4, pp. 162–164, Apr. 2003.

26. A. Belenky and N. Ansari, "Tracing multiple attackers with deterministic packet marking (DPM)," in *Proc. of IEEE PacRim*, Aug. 2003, to be published.

27. W. Feller, *An Introduction to Probability Theory and Its Applications.* John Wiley & Sons, Inc., 1968.

28. C. Shannon, D. Moore, and K. C. Claffy, "Beyond folklore: observations on fragmented traffic," *IEEE/ACM Trans. Networking*, vol. 10, no. 6, pp. 709–720, Dec. 2002.

29. J. Mogul and S. Deering, "Path MTU discovery," RFC 1191, Nov. 1990.

30. J. Postel, "Internet protocol," RFC 791, Sept. 1981.

31. D. D. Clark, "IP datagram reassembly algorithms," RFC 815, July 1982.

32. F. Baker, "Requirements for IP version 4 routers," RFC 1812, June 1995.

33. R. Braden, "Requirements for internet hosts – communication layers," RFC 1122, Oct. 1989.

34. S. McCreary and C. K. Claffy, "Trends in wide area IP traffic patterns," in *ITC Specialist Seminar.* CAIDA, 2000.

35. S. Bhattacharyya, C. Diot, and J. Jetcheva, "Pop-level and access-link-level traffic dynamics in a tier-1 POP," in *Proc. of the First ACM SIGCOMM Workshop on Internet Measurement Workshop*, 2001, pp. 39–53.

36. E. Carter, *Cisco Secure Intrusion Detection Systems*, 1st ed. Indianapolis, IN: Cisco Press, Oct. 2001.

37. V. Paxson, "An analysis of using reflectors for distributed denial-of-service attacks," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 3, pp. 38–47, July 2001.

38. J. McHug, A. Christie, and J. Allen, "Defending yourself: The role of intrusion detection systems," *IEEE Softw.*, vol. 17, no. 5, pp. 42–51, Sept./Oct. 2000.

39. T. Bass, "Intrusion detection systems and multisensor data fusion," *Communications of the ACM*, vol. 43, no. 4, pp. 99–105, Apr. 2000.

40. IBM Systems Group, "IBM TotalStorage product guide," Available on-line, Mar. 2003. [Online]. Available: http://www.ibm.com/totalstorage

41. G. Huston, "Interconnection, Peering and Settlements – Part I," *The Internet Protocol Journal*, vol. 2, no. 1, pp. 2–17, Mar. 1999.

42. G. Huston, "Interconnection, Peering and Settlements – Part II," *The Internet Protocol Journal*, vol. 2, no. 2, pp. 2–24, June 1999.

43. L. Subramanian, S. Agarwal, J. Rexford, and R. H. Katz, "Characterizing the internet hierarchy from multiple vantage points," in *Proceedings of INFOCOM 2002 Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 2, June 2002, pp. 618– 627.

44. CAIDA, "AS internet graph," Apr. 2002. [Online]. Available: http://www.caida.org/

45. A. Feldmann and J. Rexford, "IP network configuration for intradomain traffic engineering," *IEEE Network*, vol. 15, no. 5, pp. 46–57, Sept./Oct. 2001.

46. L. Gao, "On inferring autonomous system relationships in the internet," *IEEE/ACM Trans. Networking*, vol. 9, no. 6, pp. 733–745, Dec. 2001.

47. Y. Rekhter and T. Li, "Border gateway protocol 4 (BGP-4)," RFC 1771, Mar. 1995.

48. L. Gao and J. Rexford, "Stable internet routing without global coordination," *IEEE/ACM Trans. Networking*, vol. 9, no. 6, pp. 681–692, Dec. 2001.

49. CERT, Vulnarability Database, May 2003. [Online]. Available: http://www.cert.com/

50. K. Park and H. Lee, "On the Effectiveness of Route-Based Packet Filtering for Distributed DoS Attack Prevention in Power-Law Internets," in *Proc. of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, Aug. 2001, pp. 15–26.

51. S. C. Lee and C. Shields, "Challenges to automated attack traceback," *IT Professional*, vol. 4, no. 3, pp. 12–18, May/June 2002.