# 第 XII 部

# IP トレースバック・システムの研究開発

# 第12部 IP トレースバック・システムの研究開発

## 第1章　はじめに

Traceback ワーキンググループは IP Traceback などに代表されるトレースバック技術に関する基礎研究およびトレースバック技術の実用化に取り組むワーキンググループである。

今年度は、AS 間における IP トレースバック連携アーキテクチャについてまとめた研究論文、および、IP トレースバックの実運用におけるパケットの秘匿性に関する考察をまとめた研究論文をワーキンググループの活動成果としてまとめ、発表を行った。

また、2007 年度にトレースバックシステム相互接続アーキテクチャである InterTrack の実装を公開を予定しており、現在リリースに向けてソースコードの整理を行っている。

2 章では InterTrack のアーキテクチャと AS 間における IP トレースバック情報の連携方法についてまとめる。この章の内容は IEEE Symposium on Computers and Communications（ISCC'06）で発表した内容について加筆したものである。また 3 章では、InterTrack を用いた AS 間での IP トレースバック情報の交換における情報の秘匿性について考察した内容を掲載する。3 章の内容は第 2 回情報通信システムセキュリティ時限研究会で発表を行った論文を再考し加筆したものである。

## 第2章　InterTrack Architecture

### 2.1 Abstract

The difficulties of achieving an inter-domain traceback architecture come from the issues of overcoming network operation boundaries, especially the leakage of sensitive information, the violation of the administrative permission and the cooperation among Autonomous Systems (ASes). We have proposed InterTrack in [84] as an interconnection architecture for different traceback systems and other Denial of Service (DoS) attack countermeasures. In this document, we argue that only disclosing AS status to others can reconstruct the reverse AS path of an attack without the leakage of sensitive information or the violation of the administrative permission. Comparing our architecture with other traceback architectures, we also discuss the feasibility of our autonomous traceback architecture.

### 2.2 Introduction

In order to automate or expedite the manual tracking against DoS attacks or Distributed DoS (DDoS) attacks, many traceback techniques have been studied and proposed. Traceback techniques are techniques that reconstruct the attack path, and locate the attacker nodes by correcting the attack traffic, routing information, marked packets, or audit log of the attack traffic[16]. Several traceback techniques are already available as free softwares, vendor products or operation techniques within one network domain[5, 7, 15, 36, 39].

Unfortunately, no traceback techniques, which can reconstruct the attack path across several network domains, are employed or practiced in the real network operation yet. The difficulties of deploying inter-domain traceback techniques are derived from such operational issues as follows: *(A) The risk of exchanging sensitive information about the inside of each network domain.* Leakage of detailed backbone topology is a serious problem on a network operation. *(B) The fear of misuses of the traceback technique on each network domain by others.* In addition to the leakage of sensitive information, misuses of traceback systems waste resources on each AS. Furthermore, the traceback operation has been closely tied to ISP backbone network security. Arbitrary trials of traceback by unauthorized people would not be acceptable to most ISPs. *(C) The risk of depending on an unique traceback technique.* Each traceback technique has pros and cons[16]. Even if a specific inter-domain traceback technique is well deployed, attackers will develop evasion attacks sooner or later. Also, in order to run a specific inter-domain traceback technique, several ASes should deploy it at the same time. If other ASes deploy another traceback technique, operators have to contact other ASes after all. In practice, many ISPs employ multiple detection and traceback tools in their networks[140]. From the viewpoint of the current traceback operation, depending on a specific traceback technique is not practical.

These operational issues arise when a trial of traceback attempts to expand beyond the network boundaries. Many proposed traceback techniques lack or ignore the boundaries of network operation and the difference of the operational policies among different network domains. As remarked in the Arbor network's report[140], it is sure that inter-domain traceback and attack mitigation mechanisms need to be deployed ubiquitously across the Internet. Hence, an acceptable traceback architecture to ISP operators that meet operational requirements must be designed and deployed.

In order to operate the inter-domain traceback in practice, we have proposed *InterTrack* as an interconnection architecture for traceback systems to overcome the issues on the inter-domain traceback[84]. InterTrack automates recursive traceback attempts across several ASes, while at the same time allowing each AS to hide sensitive information, and to operate traceback system within its own network domain along with its operation policy.

The key ideas of InterTrack are as follows: (i) a hierarchical architecture according to the network operation boundaries, (ii) phased-tracking and the federation of internal traceback trials for the inter-domain traceback, (iii) concealment of sensitive information on exchanging traceback information among ASes, and (iv) components and APIs for independence from a specific technique, for multi-layer tracebacks and for self-defending mechanisms.

In this chapter, we present InterTrack, an autonomous architecture in order to make the inter-domain traceback practically usable. The key ideas of InterTrack are as follows:

- a hierarchical architecture according to the network operation boundaries
- phased-tracking and the federation of internal traceback trials for the inter-domain traceback
- concealment of sensitive information on exchanging traceback information among ASes
- modular components and APIs for independence from a specific technique, for multi-layer tracebacks and for self-defending mechanisms

This chapter is organized by sections. First, several assumptions on traceback are given in Section 2.3. The goals of InterTrack are discussed in Section 2.4. We also define the requirements of an inter-domain traceback architecture in Section 2.5 along with assumptions and goals. Following the overview of the architecture of InterTrack in section 2.6, we elaborate the AS path reconstruction mechanism of InterTrack in section 2.7. In section 2.8, the feasibility of InterTrack is discussed. In section 2.9, we describe the details of the prototype implementations of InterTrack. Next, in section 2.10, we evaluate the architecture and the prototype implementation. We also compare InterTrack with other inter-domain traceback architectures in section 2.11. Finally, we summarize this chapter in section 2.12.

## 2.3 Assumptions

At the beginning of the discussion about InterTrack, we assume several assumptions about

ASes on the traceback:

- Each AS has multiple tools of detection, of traceback, or of prevention for its own network domain.
- Each AS does not want to allow other people to operate or investigate its own network without permission.
- Each AS cannot investigate the inside of its customer's network without permission by the customer.
- Each AS does not want to reveal inside information to others without some reasonable procedures.
- Each AS does not want to be bothered by the traceback queries when the AS is not included in the attack path.

The first assumption is in accordance with the arbor network's security report[140] and with the fact that many implementations or techniques for the inter-domain traceback have already been made available[5, 7, 13, 15, 36, 38, 39, 40, 47, 71, 78, 81, 96, 180, 189]. The next two assumptions speak to the administrative permission to operate a network domain. The fourth assumption reflects the confidential or sensitive information on a network operation. The internal information of a network domain is likely to be secret, and if other people want to get such secrets, they would have to sign a non-disclosure agreement or file in a court procedure. The last assumption is used here because a traceback trial puts a high cost on human resources, network resources, or server resources; therefore, broadcasting a request message or additional traffic for a traceback to other ASes regardless of their status against the issued attack is undesirable.

## 2.4 The Goals of InterTrack

On designing InterTrack as a practical interconnection architecture for traceback systems or as an inter-domain traceback architecture, we accomplish several goals. These goals include:

- Detecting the upstream neighbor ASes of the attack.
- Reconstructing the reverse AS path of the attack.
- Automating the procedures to request a traceback to other network domains.
- Interconnecting any countermeasures of DDoS attacks to expedite attack protection.
- Protecting or isolating attacker nodes on an attacker-side AS with the AS own decision and operation policy.
- Achieving these five goals along with the five assumptions mentioned in section 2.3.

The first three goals are for the traceback trials on InterTrack. InterTrack employs various traceback techniques to investigate inside an AS; therefore, InterTrack must be effective in tracking attacks as a manager component for controlling and working various traceback techniques together in only one AS. Considering the leakage of sensitive information, InterTrack must reconstruct the reverse AS path instead of the actual attack path expressed by router hops. The effectiveness of aggregation router hops to AS hops have already been mentioned in [179] and [62]. InterTrack must also automate the procedures to request a traceback to other network domains. The manual operation on the procedures required to ask a traceback to other network domain spend a lot of time and manual traceback is likely to be finished before the attack finishes or the attack pattern changes. If InterTrack automates these procedures, the time spent for a traceback will be shorter.

The fourth goal of InterTrack is to cooperate with various countermeasures of DDoS attacks. Recently, several attack mitigation products work together in a single vendor environment[37], or among several vendors through a specific data format or API[166]. In order to achieve the correlation among multi-vendors' attack mitigation products, each vendor has to develop interfaces or a new protocol for other vendor products. An interface and several messages or protocols are provided by InterTrack for reducing such developing overhead.

The last two goals come from the assumptions about AS's network operation policies. InterTrack must be designed along with the assumptions described in Section 2.3.

### 2.5 Requirements

According to assumptions and goals of InterTrack, several requirements for the inter-domain traceback architecture are provided as follows:

- The architecture must leave each AS to decide to inherit a request of tracking by each AS's operational policy.
- The architecture should leave each AS to decide whether or not to investigate the inside of each own network domain more deeply. The architecture should also allow each sub-domain of an AS to decide whether or not to inspect each sub-domain's network by each sub-domain's operation policy.
- The architecture should allow each AS to take another action along with a tracking result such as a filtering or another tracking.
- The architecture should not forward request messages to ASes which have no relation to the issued attack.
- The architecture should not reveal sensitive information of an AS to others.
- A message exchanged in the architecture should have its own traceability to prove or to confirm the issuers of the message.
- The architecture should be independent from specific traceback techniques.
- The architecture should track back an attack on a dual stack environment, even when the attack employs some address translation techniques[24, 44, 234].
- The architecture should have the capability to cooperate with detection systems or protection systems.
- The architecture should exclude human beings as long as possible.

The first three requirements are essential for keeping the network operation boundaries. A routing operation reflects the contracts among ASes or the contracts between an AS and its customers. In order to keep such contracts and operational boundaries, network operators control each network domain with responsibility and cooperate with each other on the relationship of mutual trust based on such contracts. A traceback trail is a trial to confirm the routing path of unwanted traffic; thus, a traceback operation should follow the manner of other routing operation.

Next three requirements are used to block misuses of the traceback architecture. The operation of traceback will consume many resources on the related ASes; therefore, the traceback architecture should not generate or flood meaningless requests if possible. In order to reduce the damage of misuses, the message should not convey such sensitive information that might cause the leakage of secrets or confidence of an AS. Even when a misuse or a compromised action occurred, the traceability of the message will identify the offender.

The next two requirements deal with evasion attacks of each traceback technique. If the architecture depends on one specific traceback technique, attackers will develop evasion attacks and hide the location of the attacker nodes. In addition to this, many operation systems come to support the IPv4/IPv6 dual stack[4, 143], and several attacks come through a 6to4 IPv6 tunneling[256]. If the traceback architecture cannot track back attacks on the IPv6 network or attacks through some translators, the majority of attacks will shift in such a complex attack[256]. Hence, the independence from any specific traceback techniques and the independence of the versions of IP are required.

The last two requirements are supposed to automate the traceback operation. According to the taxonomy compiled by Mirkovic et al.[147], many DDoS attacks employ reflector nodes or step-stone nodes; therefore, the attacker nodes which are detected by a traceback trial might be just step-stones or reflectors. In order to detect commander nodes or true attacker nodes, the traceback architecture should cooperate with detection systems to start further traceback trials. As a matter of course, network operators will apply filters or run attack mitigation techniques after a traceback operation. To automate the process of attack mitigation, the architecture should be able to export the result of a traceback trial as a trigger of the attack mitigation. Then, an attacker may change the pattern of attack traffic to avoid the effect of such mitigating actions. Combating with changes of a complex attack, the time spent to trace an attack path should be as short as possible. Because the time spent by a person is remarkably longer than the time spent by a computer, the architecture should exclude human beings if possible. It is a challenge to construct an automated traceback procedure while network operators on each AS can control the traceback operation on its own network according to each AS's operational policy.

## 2.6 Overview of InterTrack

The main goal of InterTrack is to reconstruct the reverse AS path, which is the true attack path in AS hop level, and to detect the source ASes of an attack if possible. Another goal of InterTrack is to achieve the cooperation among traceback systems, detection systems and prevention systems inside an AS. InterTrack also aims to expedite the human operation and the cooperation among ASes on tracking an attack.

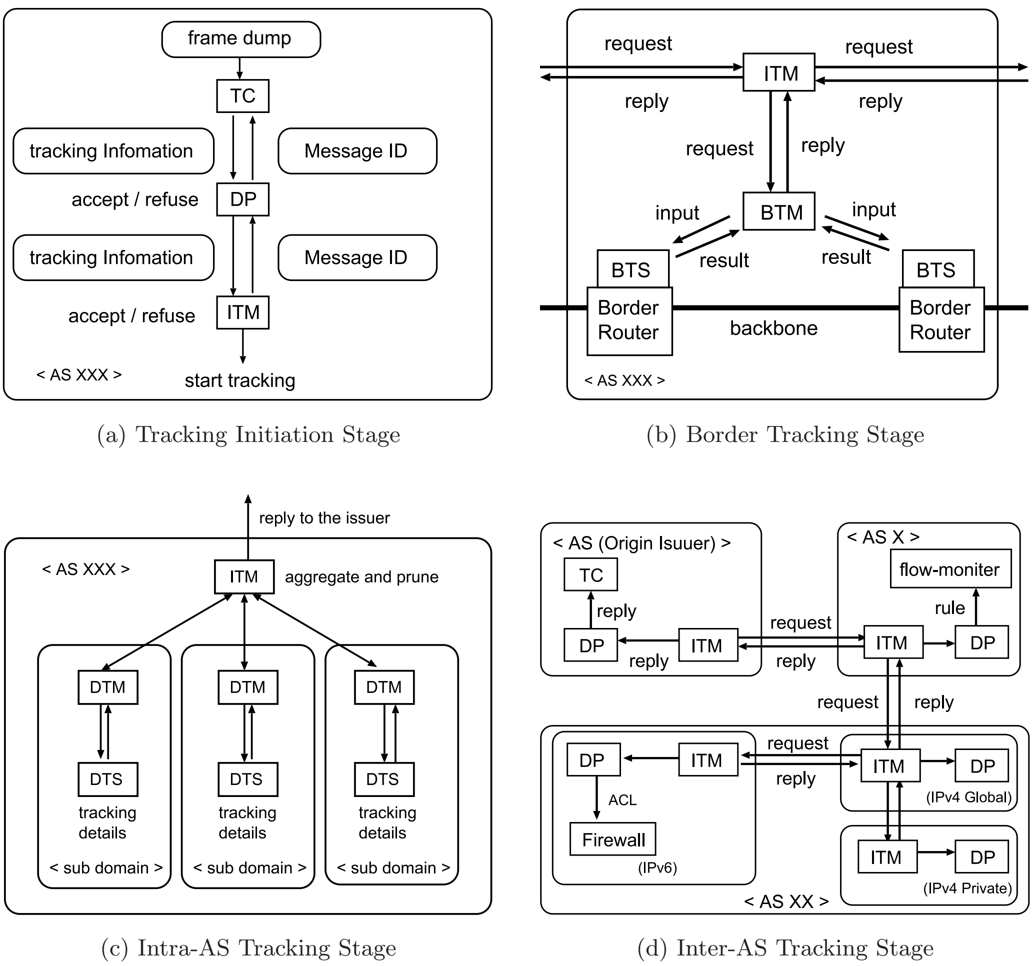
### 2.6.1 Architecture

The architecture of InterTrack refers to the Internet routing architecture. The reason why InterTrack refers to the Internet routing architecture is that the Internet routing architecture is designed along with the boundaries among operation domains and the differences of operational policies of each operation domains. For example, BGP shows the boundaries and contracts among ASes; on the other hand, OSPF sub-area can express the boundary of the different operation domain on an AS such as the boundaries among the network operation center and other departments in an enterprise network. Usually, a network operator cannot operate other network domains. On the network boundaries, network operators cooperate with each other to configure their own network equipments for achieving the proper routing. In a traceback trial, network operators cannot investigate other network domains as well as other network operations; therefore, they try to detect upstream neighbor ASes to ask them for further tracking.

In InterTrack architecture, each AS has a set of InterTrack components. A set of InterTrack components includes; the Inter-domain Tracking Manager (ITM), Border Tracking Manager (BTM), Domain Traceback Manager (DTM), Decision Point (DP), and Traceback Client (TC). Figure 2.1 shows the overview of InterTrack architecture. A phased-tracking approach is applied on inter-domain traceback trials through InterTrack. InterTrack separates a traceback trial in four stages along with network boundaries; *the tracking initiation stage* (Fig. 2.1(a)), *the border tracking stage* (Fig. 2.1(b)), *the intra-AS tracking stage* (Fig. 2.1(c)) and *the inter-AS tracking stage* (Fig. 2.1(d)). After accepting a traceback request on the tracking initiation stage, each AS preliminarily investigates its own status against the issued attack on the border tracking stage. On the border tracking, an AS judges by InterTrack whether or not the AS is suffered from an attack, whether or not the AS is forwarding malicious attack packets, or whether or not the AS is suspected of having attacker nodes on the inside. Triggered by the investigated AS status, InterTrack runs the inter-AS tracking stage and the intra-AS tracking stage in parallel.

### 2.6.2 Behaviors of InterTrack Components

An inter-domain traceback on InterTrack is composed of the federation of internal traceback trials on ASes through the phased-tracking stages. Here, we explain the characteristics of each InterTrack components and behaviors of




(a) Tracking Initiation Stage (b) Border Tracking Stage

(c) Intra-AS Tracking Stage (d) Inter-AS Tracking Stage

**Fig. 2.1.** Tracking on InterTrack

components on each tracking stage.

ITM on each AS controls traceback trials on its network domain along with the operation policy of the AS. ITM also mediates neighbor ASes to exchange the traceback information. Any traceback information from other ASes comes from the ITM network. The ITM network is an overlay network composed by a number of point-to-point peering of ITMs between two ASes. ITM network is mapped on the BGP-peering relationship so that each ITM communicates only ITMs on neighbor ASes according to the trust or the contract on the BGP-peering.

TC is an interface of InterTrack to request a traceback for network operators or detection systems such as intrusion detection systems. In the tracking initiation stage (Fig. 2.1(a)), DP authenticates and authorizes TCs; on the other hand, ITM assigns a message ID to distinguish a traceback trial on the whole InterTrack architecture. DP is a separated module function of ITM to authenticate TCs and to control request rates from a TC. Because ITM not only deals with attack requests from the inside and also treat requests from neighbor ASes, the overhead of authentication and rate limits of TC may obstruct ITM in processing other requests. Therefore, the authentication function and the rate limit function of ITM are delegated to DP.

On the border tracking stage (Fig. 2.1(b)), InterTrack preliminarily investigates the AS status expressed by the directions of the issued attack, by the possibility of the existence of an attacker on the inside of the AS, and by the information of address translation. When the result of the border tracking stage reveals the upstream neighbor ASes, InterTrack kicks off inter-AS tracking stage and propagates the traceback request message to each upstream neighbor AS through the ITM network. The neighbor ASes recursively runs the border tracking stage, and reconstruct the reverse AS path of the issued DDoS attack. Each AS adds its own AS status in the reply message and returns the reply message to the issuer neighbor AS through the ITM network. In parallel with the inter-AS traceback stage, an AS can start the intra-AS tracking stage when the result of the border tracking stage showed that the AS might have attacker nodes on the inside (Fig. 2.1(d)). The intra-AS tracking stage is the deep internal inspection on each sub-domains of the AS (Fig. 2.1(c)). The results of tracking on each sub-domain are aggregated by the ITM of the AS and stored in the AS's DP. The result of intra-AS tracking is not delivered to other ASes through InterTrack because the result of intra-AS tracking is an internal information and may include personal information such as the information of the owner whose PC was infected by some worm.

On the border tracking stage and the intra-AS tracking stage, each AS can use various traceback techniques or traceback systems according to their characteristics. Border Tracking System (BTS) is a specified traceback system to investigate the direction of the issued DDoS attack on the network boundary and judge the AS status. On the other hand, Domain Tracking System (DTS) is a traceback system for deep internal inspection on an AS. In other words, a DTS is a traceback system to locate attacker nodes logically and physically (e.g., the logical location is the nearest edge router or the incoming port on a layer 2 switch; on the other hand, the physical location is the geographical location of the nearest router or switch).

Each traceback system communicates InterTrack through a BTM or a DTM. Both BTM and DTM are wrapper components to convert the traceback request to the input values for employed traceback systems, and translate the results of traceback systems into the InterTrack message. BTM and DTM can be implemented as a module or a proxy to exchange the traceback information with the manager server of a specific traceback system.

Each InterTrack component communicates with only neighbor components through an IPsec encrypted TCP connection[121] in order to reflect the trust among network domains properly, to protect abuses or attacks from unauthorized nodes, and to keep the information of a traceback secret from others. In addition, each ITM sends heartbeat messages to neighbor components in order to confirm the existence of each neighbor component and in order to adjust the time synchronization.

### 2.7 Reverse AS Path Reconstruction

In this section, we explain the detail of the reverse AS path reconstruction and discuss the feasibility of InterTrack. InterTrack reconstructs the reverse AS path of a DDoS attack through the border tracking stage and the inter-AS tracking stage. The border tracking stage on an AS reveals the AS status against the attack. If the border tracking stage judges that the AS received the attack traffic from some of upstream neighbor ASes, an ITM forwards an ITM trace request message to those upstream neighbor ASes which may forward the attack traffic. Here, we explain the detail of AS status and the consistency of ITM trace reply message which reveals the reverse AS path to the original issuer ITM. Figs. 2.2 and 2.3 show examples of an ITM trace request message and its ITM trace reply message.

#### 2.7.1 AS Status against a DDoS Attack

An ITM decides actions by the AS status revealed on the border tracking. On the forwarding an ITM trace reply message, each ITM on the reverse AS path adds its AS status into the ITM trace reply message. The variations of AS status against a DDoS attack are shown in Fig. 2.4. On a traceback trial, an AS will have one of twelve variations of the AS status. Eight

```xml
<?xml version="1.0"?>
<ITMTraceRequest>
  <DestinationITMID>v6-65002</DestinationITMID>
  <Origin>v4-65001</Origin>
  <SequenceNumber>10000</SequenceNumber>
  <TTL>5</TTL>
    <Footmark transform="yes">
      <PacketDump iftype="0x86">XXXX XXXX XXXX XXXX </PacketDump>
      <TimeStamp><sec>1132613480</sec><usec>159368</usec></TimeStamp>
      <TransPacket>
        <Border>6TO4</Border>
        <PacketDump iftype="0x86">XXXX XXXX XXXX XXXX </PacketDump>
      </TransPacket>
    </Footmark>
    <ITMPathList>
    <Origin>v4-65001</Origin>
    <NextHop depth="1">v4-65002</NextHop>
  </ITMPathList>
</ITMTraceRequest>
```


**Fig. 2.2.** ITM trace request message


```xml
<?xml version="1.0"?>
<ITMTraceReply>
  <SourceITMID>v4-65002</SourceITMID>
  <Origin>v4-65001</Origin>
  <SequenceNumber>10000</SequenceNumber>
  <TraceResult type="FOUND">
    <ITMSubTrees>
      <ITMSubTree depth="0" type="FOUND">
        <ITMID>v4-65002</ITMID>
        <NextHops>
          <Incomings>
            <ITMID>v6-65002</ITMID>
          </Incomings>
          <Outgoings>
            <ITMID>v4-65001</ITMID>
          </Outgoings>
        </NextHops>
      </ITMSubTree>
      <ITMSubTree depth="1" type="FOUND">
        <ITMID>v6-65002</ITMID>
        <NextHops>
          <Outgoings>
            <ITMID>v4-65002</ITMID>
          </Outgoings>
        </NextHops>
      </ITMSubTree>
    </ITMSubTrees>
  </TraceResult>
</ITMTraceReply>
```


**Fig. 2.3.** ITM trace reply message

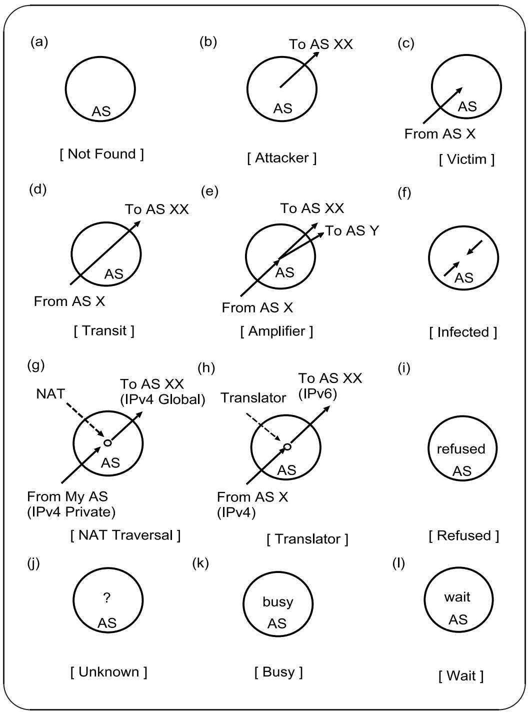



**Fig. 2.4.** Variations of state of an AS on an attack

statuses can be expressed by the combination of following informations: directions of the forwarding path of an attack, the necessity of the detailed internal inspection, the notification of an address translation. In addition to these combinations,

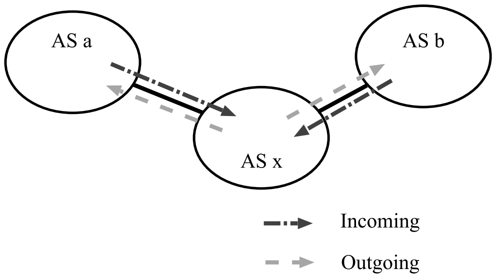



**Fig. 2.5.** Directions of traffic on an AS

there are four error statues.

First, we consider the relation with a neighbor AS. Basically, an AS status is composed of the status of the inside and each status on the point to point (P2P) link between each neighbor AS. The status on a P2P link can be described by the packet directions, that is, *incoming* and *outgoing* (Fig. 2.5).

If an AS find a packet or flow of the issued DDoS attack on the incoming direction in a P2P link, the P2P link is a *Victim* link, that is, the AS receives the DDoS attack flow from the neighbor AS. The status of a P2P link is *Attack* when the DDoS attack is detected on the outgoing direction on the P2P link. When the DDoS attack is not detected either on the incoming direction or the outgoing direction of an P2P link, the AS status on the P2P link is *Negative*. If the DDoS attack is found both on incoming and on outgoing direction of a P2P link, it indicates a *Loop* has occurred. Usually, *Loop* is an error state which indicates the error of the BTS or some wrong routing so that a more detailed investigation is required. Therefore, a *Loop* link is defined as equal to the *Attack* and *Victim* link.

Next, we consider about the AS status with all relations between each neighbor AS. If all P2P links shows *Negative*, the AS status is judged as *Not Found* (Fig. 2.4(a)). When each border tracking on each P2P link judges either *Negative* or *Victim*, an AS is in *Victim* state (Fig. 2.4(b)). On the other hand, an AS has *Attacker* state when the results of P2P links either *Attack* or *Negative* (Fig. 2.4(c)). An AS status is judged as *Transit* when the investigation results of P2P links contains *Negative*, *Attack*, and *Victim* (Fig. 2.4(d)).

Here, we consider the AS status with the internal status of an AS. If there is a possibility that an attacker node is inside the AS, the AS is in *Amplifier* state (Fig. 2.4(e)). The border tracking stage judges the AS status as an *Amplifier* state in the following cases:

- The number of *Attack* links is more than the number of *Victim* links.
- The amount of traffic to some *Attack* link is increased remarkably.
- One of P2P links shows *Loop*.
- When the AS connects to a neighbor AS with several links, one P2P link is *attack* and the other is *Victim*.

When an AS is in the *Amplifier* state or in the *Attacker* state, the AS has to start the intra-AS tracking. If there is a need to start intra-AS tracking, a BTM adds an *ASK_DTM* flag in the reply message to the ITM. A BTM also adds an *ASK_DTM* flag when the border tracking stage shows the *Infected* state (Fig. 2.4(f)). The *Infected* state indicates that the AS is attacked not from other ASes, but from the inside.

Some attacks employ address translation techniques, such as IPv4/IPv6 tunneling or NAT[256]. If an attack comes from the IPv4 private segment which is operated by an AS, the border tracking judges the AS is in a *NAT traversal* state (Fig. 2.4(g)). The border tracking stage indicates a *Translator* state when the attack comes through an 4to6 tunnel or 6to4 tunnel (Fig. 2.4(h)). In these translation cases, a BTM adds the information of the translation. If an AS employs a logging technique such as SPIE[15] for the BTS, the BTS may store the previous packet information before the packet was translated. If BTS has the previous packet information, then the BTM also adds such previous packet information into the reply message. When an ITM receives the information of the translation from the BTM, the ITM adds the information into the ITM trace request message. Then, in order to start further traceback trials in different address space, the ITM changes the role to the ITM on another address space, or forwards ITM trace request message to another ITM on the same AS.

Finally, we consider four error states on the border tracking. Each AS can refuse a traceback request along with its operational policy. If an AS refuses a traceback request, then, the ITM adds a *Refused* state as its AS status into the ITM trace reply message, and sends the ITM trace reply message to the issuer neighbor ITM (Fig. 2.4(i)). If an ITM, a BTM, or a BTS is busy because of processing other traceback requests, the AS status becomes *Busy* (Fig. 2.4(j)). When something wrong has occurred in the border tracking stage, the BTM replies with some error message. Then, the AS status is judged as *Unknown*, the ITM adds an error message from the BTM into the ITM trace reply message (Fig. 2.4(k)). An ITM will reply *Wait* as the AS status if an AS is in the border tracking stage, but the AS needs much more time to get the result due to the limitation of the BTS (Fig. 2.4(l)).

### 2.7.2 Loop Detection on Forwarding an ITM Trace Request Message

In order to avoid the loop in forwarding an ITM trace request message, the ITM on each adds its *ITM ID* into the *ITM Path List* field on the ITM trace request message when the ITM forwards the ITM trace request message to neighbor ASes. Using the ITM path list and the message ID of an ITM trace request message, each ITM judges whether a loop on the message forwarding occurs or not.

An ITM has ITM IDs which represent the address spaces covered by the ITM. The ITM ID on each address space is represented by the address space information and AS number. For example, each ITM ID of AS 2500 is `v4-2500` in the IPv4 global network, `v4-2500-private` in the IPv4 private network, `v6-2500` in the IPv6 network. The origin ITM, which originates an ITM trace request message, assigns a message ID to a traceback request on the tracking initiation stage. A message ID is composed of the ITM ID of the origin ITM and the sequence number assigned by the origin ITM.

When a result of the border tracking contains a neighbor AS as the upstream neighbor and the ITM ID of the neighbor AS is already included in the ITM path list field, then, an ITM concludes a loop occurs and the ITM does not forward the ITM trace request to the neighbor ITM. An ITM also judges a loop when the received ITM trace request message contains its own ITM ID in the ITM path list field. In this case, the ITM does not reply with the ITM trace reply message to the issuer neighbor ITM. The ITM path list field contains all ITM IDs which represent the partial reverse AS path of the issued attack. Each ITM ID indicates each traversal AS of the ITM trace request; therefore, an AS can refuse the traceback request when an untrusted AS is included in the ITM path list.

In addition to the ITM path list field, an ITM trace request message contains Time-To-Live (TTL) field in order to stop an endless forwarding. Each ITM decrements the value of the TTL field on forwarding an ITM trace request message. If the TTL value reaches zero or a negative value, an ITM stops forwarding the ITM trace request. According to the analysis by team Cymru[248] or by Geoff Huston[91], the observed maximum AS path length was less than 40 hops, and the weekly average AS hops is about 5 hops. Therefore, the maximum TTL is settled in 64 with some provisioning and the default TTL value is defined as 5.

### 2.7.3 Inconsistency among Tracking Results of each AS

An ITM trace reply message may contain some inconsistencies among each ITM result on the Reverse AS path. The inconsistency will occur in these cases as follows: When the ITM trace reply message reports *Victim* or *Infected* state on the result of other ITM, this inconsistency is caused by two kinds of mistakes. One case is that the BTM, whose AS is in several hops away from the origin ITM, mis-judged the AS status as *Victim* or *Infected*. The other inconsistency is that an ITM request message was generated and forwarded from a transit AS and the ITM request message wrongly reached the true *Victim* AS by the mis-judges in several ASes.

If the ITM trace reply message contains *Not Found* on the results of other ASes, an ITM made an incorrect forwarding to neighbors that are not in the real attack path, or the BTM of an AS mis-judged the AS state as *Not Found*. The ITM Subtree does not have the ITM ID of the former issuer on the outgoings field when the result of the border tracking on the deeper hop AS was wrong, or when the mis-forwarding continuously occurred in several hops.

The ITM trace reply message may contain a loop in the *ITM Subtrees* field. If the same ITM ID is listed both on the Outgoings field and the Incomings field of a ITM Subtree field, the ITM Subtree would indicate that the border tracking on the AS was wrong or the routing loop really occurred between two ASes. When a shallower hop ITM's ID is contained in the Incomings field on the result by the deeper hop ITM, the BTM on the deeper hop AS mis-judged, or the wrong message forwarding continuously occurred from the shallower hop AS to the deeper hop AS.

Even when an ITM trace reply message contains some inconsistency, the network operator on the original issuer AS can contact each AS by referring to the reverse AS path on the ITM trace reply message. On the other hand, each AS on the reverse AS path holds the partial ITM paths both on the received ITM trace request messages and on the ITM trace reply messages from neighbor ITMs. Hence, each AS can confirm the inconsistency on the reverse AS path by itself even when the AS is not the original issuer AS.

### 2.7.4 Analysis of Attack Cases against the InterTrack

Attack cases to the InterTrack architecture are considered here, and the feasibility of InterTrack against each attack case is then discussed, while the defending techniques to cover the vulnerabilities of InterTrack are explored.

Against the Numerous requests from a TC, DP can limit or drop requests from TC in a unit time. In addition to this, DP's authentication and rate-limit functions can be separated from ITM; therefore, the ITM network cannot be affected by the processing rate-limits on DP and can treat other requests while a TC attacks by numerous requests.

An attacker may try to hijack a TCP connection between components. In InterTrack, each TCP session between two components is based on IPSec authentication[121]; therefore, each component will not accept the request of connection from an unauthorized node. Moreover, network operators can easily apply filter rules on the nearest routers for each component in the source address and the destination address pair, because the neighbor components of each component are fixed or limited. Also, the IP address of a component should be known by only neighbor components, hence, it is not necessary to assign an FQDN to the IP address of a component and to propagate the FQDN by DNS. Therefore, network operators can hide the IP addresses of InterTrack components from unauthorized people. Furthermore, if an ITM has several interfaces, each connection to a neighbor component can be achieved on a closed private network. By a combination of such techniques, network operators can protect InterTrack from SYN floods, or UDP floods, even when the attacker spoofs the source IP address of attack packets as the IP address of a component.

Since each connection of two components can be constructed on the closed private network, such a private network is not affected by the bandwidth consumption on the main link of a neighbor AS. Even when the connection of two components is achieved as the on-line connection, priority queuing techniques can turn down the effect of the bandwidth consumption against the InterTrack messaging. Also, the ITM network is an overlay network. If other routing paths are prepared, the ITM network can tolerate the influence of link-downs or route flaps.

If an ITM is hijacked by an attacker, it becomes a serious security issue over several ASes because the attacker can steal the inside information of the hijacked ITM's AS and can attack by faked ITM trace request messages and spoofed ITM race reply messages. By hijacking other InterTrack components, an attacker can steal the inside information or can dirty traceback results. Therefore, each AS should protect the intrusion to the InterTrack as strongly as possible.

When an ITM is hijacked by an attacker, the hijacked ITM may send a faked ITM trace reply message which contains a spoofed reverse AS path. In this case, if the network operator on the original issuer AS contacts all ASes on the spoofed reverse AS path to verify the reverse attack path, then, the network operator on each AS will find the inconsistency on the same traceback trial and will detect the hijacked AS.

## 2.8 Discussion

### 2.8.1 A Multi-Layer Traceback for Complex Attacks

Considering DDoS tools[147], the attacker nodes on the Attacker AS are just the step-stone nodes or the reflector nodes. In order to detect commander nodes or the PC of an attacker, InterTrack tracks the attack in multi-layers by the cooperation of detections systems and continuous trials of traceback. After the intra-AS tracking stage (Fig. 2.1(c)), the result of the intra-AS tracking is stored in DP. According to the configuration or AS's policy, DP exports the result to detection systems to correct the command packets or the pre-reflected packets (Fig. 2.1(d)). If a detection system catches these command packets or pre-reflected packets, it starts another traceback to the source of these packets. Proceeding this process recursively, InterTrack can track an attack on the layer 7 networks, that is, traces back the attack from the step-stone node to the *the true source* of the attack.

This continuous traceback trial is used to track the attacker node on the inside of an AS. When an *Infected* status AS requests a traceback, the ITM of the AS starts the intra-domain tracking stage soon after. The purpose of the intra-AS tracking stage is not only revealing the reverse path on the layer 3 network but also reconstructing a reverse path on a layer 2 network and locating the attacker node. There are several traceback techniques on a layer 2 network[7, 36, 75, 85, 180], and most of them require not the IP datagram of the issued attack but the Ethernet header to get the source MAC address or VLAN information as a key of tracing. In order to track on a layer 2 network in the *Infected* case, we design the *PacketDump* field of the trace request message to include the packet payload with the Ethernet header. The process of the traceback on layer networks as follows: First, the operator or a detection system sends a whole Ethernet frame to InterTrack through a TC. After judged *Infected*, the ITM kicks the intra-AS tracking stage and finds the nearest layer 3 gateway of the attacker node. If the attacker node is in the same layer 2 network of the victim node, then DTM runs a layer 2 traceback technique and tracks the location of the attacker node. When the attacker node is in another local subnet, the DTM explores the layer 2 subnet to detect the logical and physical location of the attacker node if the DTS on the attacker's subnet stores the source MAC address of the issued packet. If not, the DP of the AS exports the result of the initial traceback to the detection system on the attacker's subnet. And then, the detection system triggers another tracking by catching an Ethernet frame of the issued attack to get the source MAC address.

### 2.8.2 Privacy Issues

In order to start a traceback trial, a TC has to input the whole Ethernet frame of the issued attack packet. Then, the privacy issue about the packet payload comes into question. Focusing on the DDoS attack, the packet payload is usually meaningless information or the binary of a malicious code; therefore, such packets do not include any personal information. A single packet attack such as SQL slammer does not have any personal information either. Hence, the privacy issue of the packet payload of a trace request will not matter as long as InterTrack is employed for the traceback of attacks. Of course, if someone tries to track other service traffic by InterTrack, the privacy issue comes to a head.

InterTrack can trace an attack in the multi-layers. Running a multi-layer traceback requires some auditing systems which store the layer 7/application level audit log or bind a MAC address to an IP datagram as its source. Using such audit systems, privacy issues of the audit log must be considered.

#### 2.8.3 Certification on InterTrack Components

Each component of InterTrack achieves IPSec encapsulated TCP sessions[121] with all neighbor components; consequently, the overhead of exchange shared keys or certificate files among ASes must be considered. The APNIC has a trial of certification of IP addresses and ASes[90] by X.509 Extensions for IP Addresses and AS Identifiers[134]. These X.509 Extensions are expected to feed into sBGP[14] or soBGP[270]. The certificate files for sBGP or soBGP can be used for the IPsec Encapsulating Security Payload (ESP)[121] between each neighbor InterTrack components as well as the IPSec ESP among components of SPIE[15].

The prototype design of the ITM trace request message shown in Fig. 2.2 is simple and it does not include the digital signatures of traversed ASes. The digital signature is effective for proving or verifying the issuer AS. The InterTrack messages are designed in XML format; therefore, an extension of messages to include such digital signature can be easily developed by adding a sub-element in the ITM path list field. The extension for the digital signatures is part of future research with the public key sharing problem mentioned above.

### 2.9 A Prototype Implementation of InterTrack

This section shows the detail of the prototype implementation of InterTrack. The basic functions of InterTrack have been developed in C language on FreeBSD, except for the function applying the AS's operational policy and IPSec encryption. LIBXML2[259] was employed for the XML parser of InterTrack messages. We also developed a sample BTM implementation and a sample DTM implementation as a proxy for PAFFI[96]. PAFFI[96] is an implementation of hash-digest-logging traceback method by Yokogawa Electric Corporation. The volume of the prototype code was about 50,000 lines totally.

#### 2.9.1 Library and InterTrack Components

We developed a library which contains common APIs used by each InterTrack components. Fig. 2.6 shows the software architecture of InterTrack. Basically, each InterTrack component processes XML messages; therefore, each component uses common APIs on message processing, on reading or writing messages, on serializing InterTrack messages, on exchanging heartbeat messages, or on managing connections between neighbor nodes. In this software architecture, each InterTrack component is modularized. Each component can be implemented by changing the algorithm of the traceback module. The characteristics of each part in Fig. 2.6 are as follows:

- *Connection*
  This part manages each connection between neighbor nodes. According to the configuration file or changes through command line, ConnectionManager connects or accepts a TCP session to a neighbor node. The DispatchLoop pools file descriptors or sockets and calls ReaderWriter to process each buffer stream or message. ThreadManager controls the mechanism or

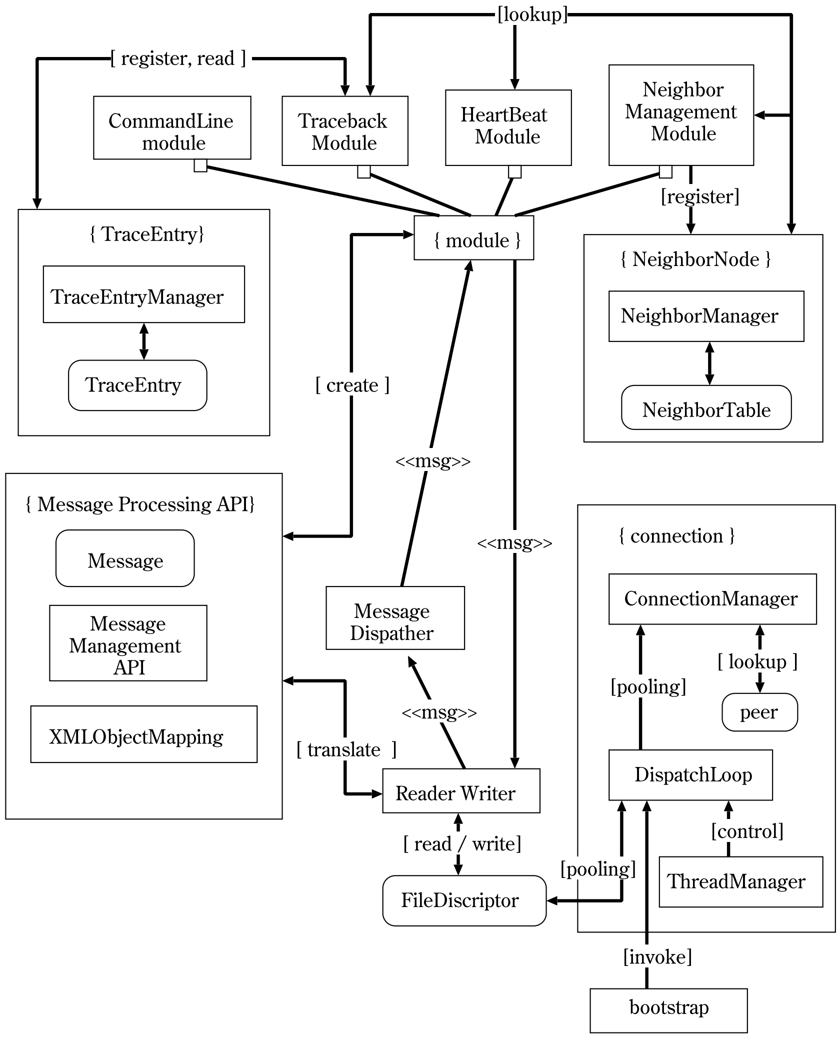



**Fig. 2.6.** The software architecture of InterTrack

policy on the DispatchLoop. We have developed only select-based DispatchLoop, however, a DispatchLoop implementation using thread could be easily implemented in the ThreadManager.

- *ReaderWriter*
  ReaderWriter controls network I/O or file I/O. Through Message Processing APIs, ReaderWriter passes InterTrack messages to MessageDispather or writes InterTrack messages into socket file descriptors.
- *Message Processing API*
  We prepared several APIs to manipulate InterTrack messages. XMLObjectMapping maps an InterTrack message in an XML string to an internal structure for InterTrack messages. We defined WITMSG as the internal structure for InterTrack messages shown as Fig. 2.7. Message Management API is a group of APIs to manipulate an InterTrack message on WITMSG structure.
- *Message Dispatcher*
  Message Dispatcher dispatches or discards messages according to the types of messages and the type of traceback module.
- *NeighborNode*
  NeighborMannager manages the entries of the NeighborTable. The NeighborTable contains the information neighbor nodes which are defined by configuration files or new configuration through command lines. The Traceback module or the Heartbeat module

```
typedef struct witmsg{
    MSG_TYPE msg_type;
    union{
        heartbeatReq    *hbreq;
        heartbeatRep    *hbrep;
        heartbeatErr    *hberr;
        clientTraceReq  *clientTraceReq;
        clientSeqRep    *clientSeqRep;
        clientTraceRep  *clientTraceRep;
        dpointTraceReq  *dpointTraceReq;
        dpointSeqRep    *dpointSeqRep;
        dpointTraceRep  *dpointTraceRep;
        itmTraceReq     *itmTraceReq;
        itmTraceRep     *itmTraceRep;
        btmTraceReq     *btmTraceReq;
        btmTraceRep     *btmTraceRep;
        dtmTraceReq     *dtmTraceReq;
        dtmTraceRep     *dtmTraceRep;
    } msg;
}WITMSG;
```

**Fig. 2.7.** WITMSG structure

looks up the information of neighbor nodes through APIs of the NeighborManager.

- *TraceEntry*

  TraceEntry is a table used to arrange several InterTrack messages in one traceback trial. The Traceback module registers a new InterTrack message, looks up an existing TraceEntry, or removes an old TraceEntry through the TraceEntryManager.
- *Module*

  The algorithm parts of each InterTrack component were modularized. When developing a BTM or a DTM, developers make the algorithm and combine the library of a specific traceback implementation on this part.
- *bootstrap*

  The Bootstrap initializes each part and starts an InterTrack component as a daemon.

Each InterTrack component was developed as a daemon; *itmd*, *btmd*, *dtmd*, *dpointd*, and *witclient*. As a sample TC, *packetcapture* was developed, which captures packets by PCAP library and passes the captured packets to *witclient* through a ring buffer on the shared memory.

### 2.9.2 Sample BTM and DTM Using PAFFI

Besides the library and daemons, a sample BTM implementation and a sample DTM implementation using PAFFI[96] were developed. The PAFFI architecture contains one manager node (PAFFI Manager) and several capture point nodes (Footmakers) which record captured packets in a Bloom filter[20]. Each Footmarker has several Bloom filters. Each Bloom filter, called a *capture point*, is mapped with an interface or a MAC address filter rule; therefore, a Footmarker can distinguish incoming traffic and outgoing traffic according to the identifier of each capture point. Hence, PAFFI can be used as BTS which can reply the variations of AS status described in section 2.7.1. Fig. 2.8 shows a sample topology when PAFFI is used as BTS.

In sample BTM/DTM implementations for PAFFI, we used the proxy type implementation style, that is, both BTM and DTM behave as clients of a PAFFI manager and translate from an InterTrack trace request message to the PAFFI request message or a PAFFI reply message to an InterTrack trace reply message. The messages between the PAFFI manager and its client are described in XML, and client sends and receives messages over HTTP[73]. Unfortunately, a PAFFI reply message does not contain AS-number field in order to indicate an upstream neighbor AS; therefore, we prepared AS mapping table on the BTM implementation. The components of the AS mapping table can be described in Table 2.1. Using this AS mapping table, BTM converts a capture point ID to the AS number of a neighbor AS and the direction of the issued packet.

## 2.10 Preliminary Evaluation

### 2.10.1 Expected Round Trip Time of an ITM Trace Request

With InterTrack, users will wonder how much time will be spent to get a reverse AS path. Here, we consider the round trip time (RTT) of an ITM trace request. For the sake of

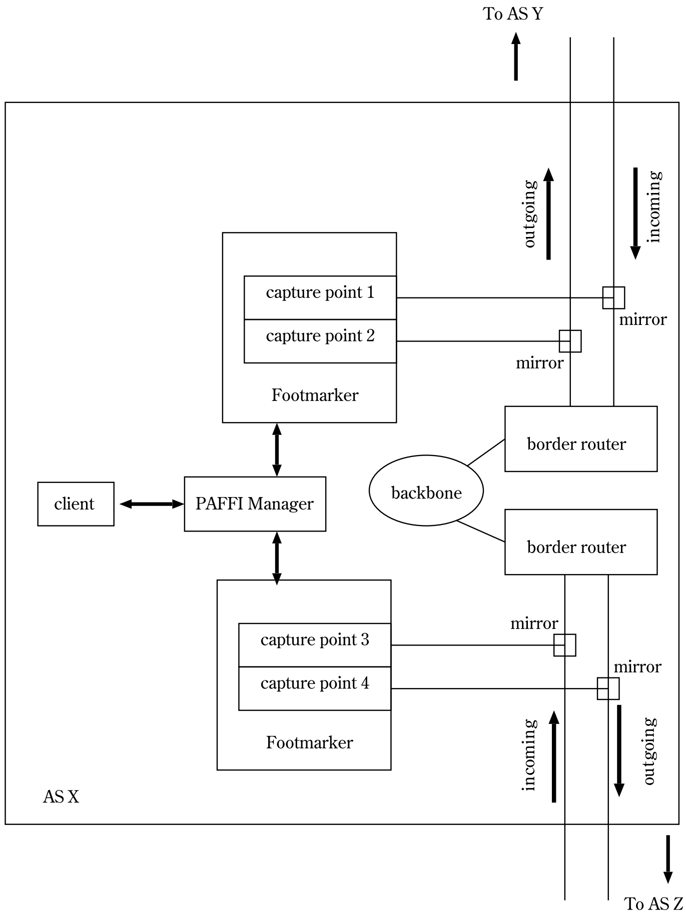



**Fig. 2.8.** The topology of PAFFI as BTS

**Table 2.1.** AS mapping table on BTM for PAFFI in accordance with Fig. 2.8

| Capture Point ID | AS number | direction |
|---|---|---|
| capture point 1 | Y | incoming |
| capture point 2 | Y | outgoing |
| capture point 3 | Z | incoming |
| capture point 4 | Z | outgoing |

simplicity, we assumed that each AS employs a hash-digest-logging method as the BTS on each network. Because of the characteristic of hash-digest-logging method, the reverse AS path always becomes a liner topology.

If an ITM trace request message is forwarded from Origin issuer AS to the AS in $n$ hops, and the AS in $n$ hops is the Attack state, suppose each parameters as follows:

- $t_{dec(i)}$: the time from when AS $i$ received an ITM trace request to when the AS $i$ decides to start the border tracking stage or to refuse the request.
- $t_{btm(i)}$: the time spent for border tracking stage
- $t_{req(i)}$: the time spent to forward an ITM request message to neighbor ASes
- $t_{wait(i)}$: the time spent to wait for all neighbor ASes to return each ITM reply message
- $t_{rep(i)}$: the time spent to make and send an ITM trace reply message
- $t_{out(i)}$: the time out threshold of $t_{wait(i)}$
- $t_{rtt(i)}$: the RTT of a traceback, that is, from the time when AS $i$ receives an ITM trace request to the time when the AS finished sending the corresponding ITM trace reply message.

Then, the maximum RTT and the minimum RTT on the AS $i$ are

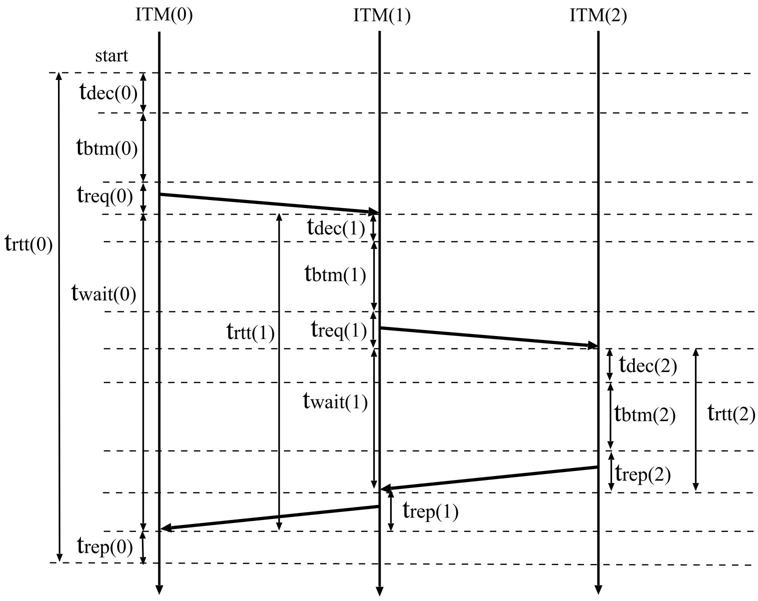



**Fig. 2.9.** The round trip time of a response of an ITM trace request with 3 ASes

$$t_{max(i)} = t_{dec(i)} + t_{btm(i)} + t_{req(i)} + t_{out(i)} \quad (1)$$

$$t_{min(i)} = t_{dec(i)} + t_{rep(i)} \quad (2)$$

$$t_{min(i)} \leq t_{rtt(i)} \leq t_{max(i)} \quad (3)$$

$$t_{rtt(i)} = t_{dec(i)} + t_{btm(i)} + t_{req(i)} + t_{wait(i)} + t_{rep(i)} \quad (4)$$

On the original issuer AS ($i = 0$), the RTT is

$$t_{rtt(0)} = \sum_{i=0}^{n} (t_{dec(i)} + t_{btm(i)} + t_{req(i)} + t_{rep(i)}) \quad (where\ t_{req(n)} = 0) \quad (5)$$

Next, we consider the expectation of RTT on AS $i$. Suppose the probabilities on each decision of AS $i$ as follows:

- $p_{dec(i)}$: the probability with which AS $i$ decides to start the border tracking stage.
- $p_{req(i)}$: the probability with which AS $i$ decides to forward the ITM trace request to upstream neighbor ASes as the result of the border tracking stage.
- $p_{out(i)}$: the probability with which the time out on the waiting ITM trace reply messages occurs.

Here, the probability that AS $i$ receives an ITM trace request messages from all neighbor ASes is described as:

$$p_{wait(i)} = p_{dec(i)}\, p_{req(i)}\, (1 - p_{out(i)}) \quad (6)$$

The expectation of RTT in the original issuer AS ($i = 0$) is

$$E(t_{rtt(0)}) = \sum_{i=0}^{n} \big(t_{min(i)} + t_{btm(i)}\, p_{dec(i)} + (t_{req(i)} + t_{out(i)})\, p_{dec(i)}\, p_{req(i)}\big) (\Pi_{k=0,\, i}\, p_{wait(k)}) \quad (where\ t_{req(n)} = 0) \quad (7)$$

Next, we consider the RTT with the false positive rate and the false negative rate on the border tracking stage. The probabilities of mis-detection of the border tracking stage are defined as follows:

1. $p_{bfp(i)}$: the false positive rate of the border tracking stage. Here, AS $i$ does not have an upstream neighbor AS on the attack path, but the border tracking stage mis-judges and indicates an upstream AS.
2. $p_{bfn(i)}$: the false negative rate of the border tracking stage. Here, AS $i$ has an upstream neighbor AS on the attack path, but the border tracking stage mis-judges and does not indicate an upstream AS.
3. $p_{btp(i)}$: the true positive rate of the border tracking stage. Here, AS $i$ has an upstream neighbor AS on the attackpath, and the border tracking stage properly judges and indicates an upstream AS.

4. $p_{btn(i)}$: the true negative rate of the border tracking stage. Here, AS $i$ does not have an upstream neighbor AS in the attack path, and the border tracking stage judges properly and does not indicate an upstream AS.

$$p_{btp(i)} + p_{btn(i)} + p_{bfp(i)} + p_{bfn(i)} = 1 \tag{8}$$

$$p_{req(i)} = p_{bfp(i)} + p_{btp(i)} \tag{9}$$

$$\begin{aligned} p_{fwd(i)} &= p_{dec(i)}\, p_{req(i)} \\ &= p_{dec(i)}\,(p_{bfp(i)} + p_{btp(i)}) \end{aligned} \tag{10}$$

$$p_{wait(i)} = p_{dec(i)}\,(p_{bfp(i)} + p_{btp(i)})(1 - p_{out(i)}) \tag{11}$$

$$p_{err(i)} = p_{dec(i)}\,(p_{bfp(i)} + p_{btp(i)})\,p_{out(i)} \tag{12}$$

$$\begin{aligned} E(t_{rtt(0)}) = \sum_{i=0}^{n} \big(& t_{min(i)} + t_{btm(i)}\, p_{dec(i)} \\ & + (t_{req(i)} + t_{out(i)})\, p_{dec(i)} (p_{bfp(i)} + p_{btp(i)})\big) \\ & (\Pi_{k=0,i}\, p_{wait(k)}) \\ & (where\ p_{wait(n)} = 0,\ p_{req(n)} = 0) \end{aligned} \tag{13}$$

### 2.10.2 Preliminary Experiments with Implementation

To evaluate the basic workloads of the ITM network, a preliminary experiment was conducted with the prototype implementation of InterTrack. For the experimental environment, 9 Dell Power Edge 1855 blade servers were used as a testbed network. The physical topology is shown in Fig. 2.10. Each blade server equipped with a Pentium III 1.4 GHz CPU, a 1024 MB RAM, and two gigabit Ethernet interfaces. These 9 blade servers were divided into two blade cages and interconnected with each other through the two layer 2 switches equipped on each blade cage. In this experiment, we measured the overhead on the message processing of ITM. A blade server was regarded as an AS, an ITM on each blade server was run with a dummy BTM function which judged all neighbor ITMs except the issuer one as upstream neighbors. On the *Control Network*, each ITM synchronized its own time to the NTP server and other ITM nodes. On the *Experiment Network*, each ITM connected to only neighbor ITMs in a liner topology according to the assumption that each AS employed a hash-digest-logging method. We measured the relation

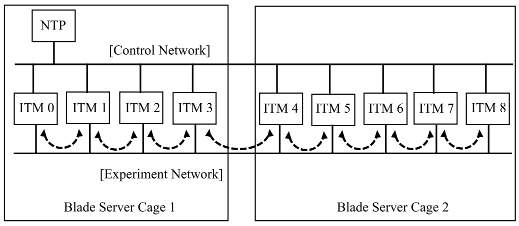



**Fig. 2.10.** Testbed topology

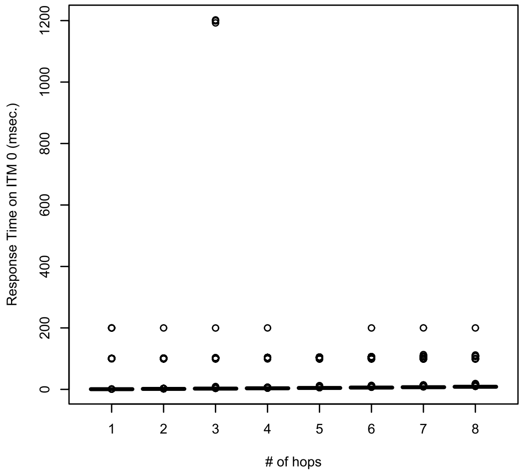



**Fig. 2.11.** RTT of an ITM trace request in a liner topology

between the RTT of ITM trace request messages on the origin issuer ITM and the number of hops forwarding the ITM trace request messages. With over-provisioning about the average length of AS hops[91, 248], we measured RTTs of the 1,000 ITM trace requests which were forwarded from 1 hop to 8 hops. Each trial ran independently with 5 second intervals. The average RTT of ICMP on the *Experiment Network* between each neighbor node was 0.197 milliseconds.

Fig. 2.11 shows the box-whisker plot of the experiments. Including outliers, all messages returned to the original issuer ITM within 1.2 seconds, and most were less than 200 milliseconds. Table 2.2 shows the data of the components on Fig. 2.11, Fig. 2.12, and Fig. 2.13. Because the value of the mean is higher than the 90th percentile on each column, each column of Fig. 2.11 draws a positive skew like Fig. 2.12.

In order to analyze the trend of the curve more deeply, the variations of RTT which were less than

**Table 2.2.** The data of box-whisker plot on Fig. 2.11 and Fig. 2.13

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| max. (msec.) | 199.939 | 199.956 | 1202.139 | 200.032 | 105.224 | 199.9 | 200.141 | 199.968 |
| mean (msec.) | 10.400 | 8.7806 | 13.155 | 12.604 | 12.124 | 13.348 | 14.903 | 14.887 |
| 90th percentile (msec.) | 1.0180 | 2.0300 | 3.173 | 4.1960 | 5.4410 | 6.9200 | 8.1240 | 9.6870 |
| 75th percentile (msec.) | 0.8125 | 1.7215 | 2.717 | 3.7720 | 4.8670 | 6.1725 | 7.4320 | 8.8835 |
| median (msec.) | 0.7025 | 1.5890 | 2.542 | 3.5815 | 4.6245 | 5.8940 | 7.1310 | 8.5300 |
| 25th percentile (msec.) | 0.6750 | 1.5120 | 2.406 | 3.4265 | 4.4650 | 5.6530 | 6.9295 | 8.2650 |
| min. (msec.) | 0.5330 | 1.3360 | 2.131 | 2.9930 | 4.0200 | 5.1790 | 6.4980 | 7.7410 |
| variance | 931.24 | 679.67 | 4891.3 | 831.22 | 672.28 | 689.99 | 715.65 | 571.68 |

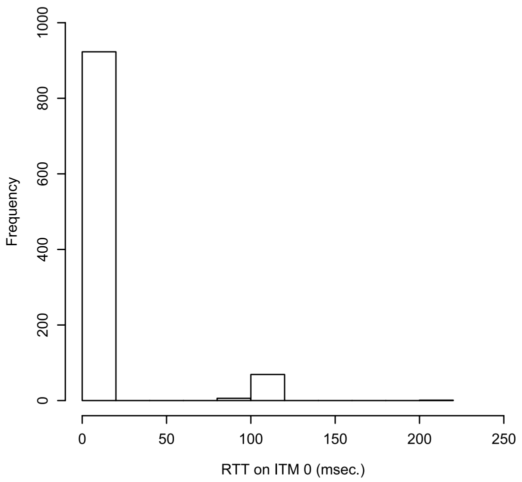



**Fig. 2.12.** Histogram of RTT on ITM 0 in a 9 hops length topology

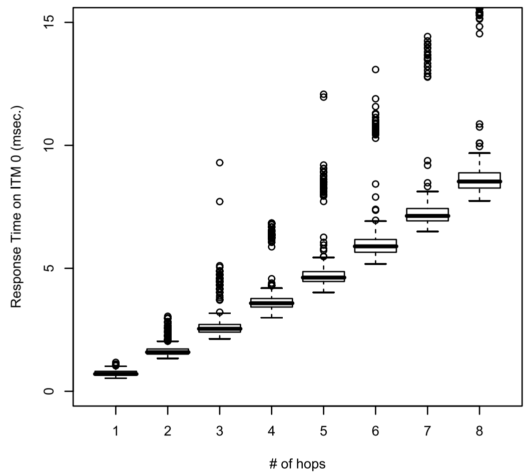



**Fig. 2.13.** RTT of an ITM trace request (scope on the box)

15 milliseconds were plotted in Fig. 2.13. The distribution of RTT on each hop length was folded in a narrow box, but the outliers of each column make the variance high. Fig. 2.13 focused on the values between the 10th percentile and the 90th percentile. The boxes in Fig. 2.13 show the curve of an increasing function.

Figs. 2.14 and 2.15 draw the distribution of the RTT on each ITM in the 9 hop length topology. The tendency of the distribution was decreased along with the distance from the original issuer ITM as in the formula described in Section 2.10.1. Fig. 2.16 shows the distribution of the processing time on the border tracking stage in the 9 hop length topology. The distribution expresses the time from receiving a request to forwarding the request to the upstream neighbor (i.e., $t_{dec(i)} + t_{btm(i)} + t_{req(i)}$). In this experiment, the tracking initiation stage on the original issuer ITM (ITM 0) was cut, that is, $t_{dec(i)} = 0$; therefore, the distribution on the ITM 0 was lower than other transit ITMs (ITM 1 to ITM 7). On the other hand, the ITM 8 was seen as the attacker AS. Because the attacker AS does not forward the request further upstream, the $t_{req(8)}$ is zero. For this reason, the distribution on the ITM 8 is lower than that of

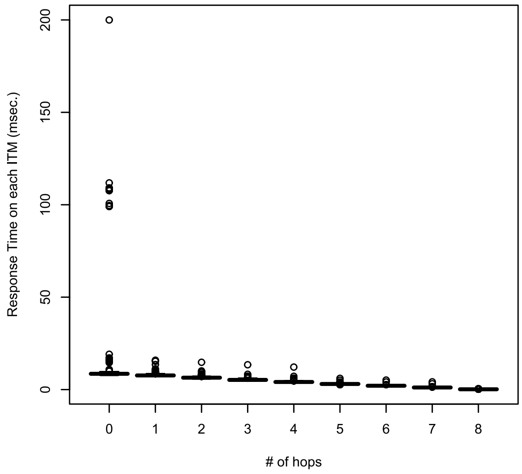



**Fig. 2.14.** RTT on each ITM in a 9 hops length topology

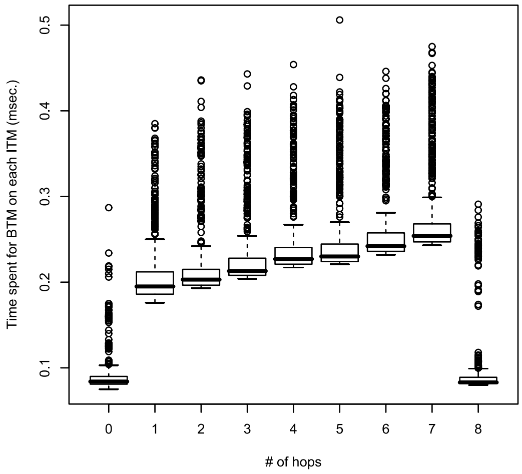



**Fig. 2.16.** The processing time of the border tracking stage

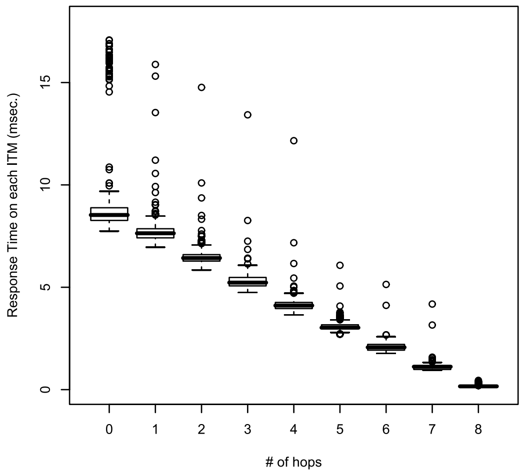



**Fig. 2.15.** RTT on each ITM in a 9 hops length topology (scope on the box)

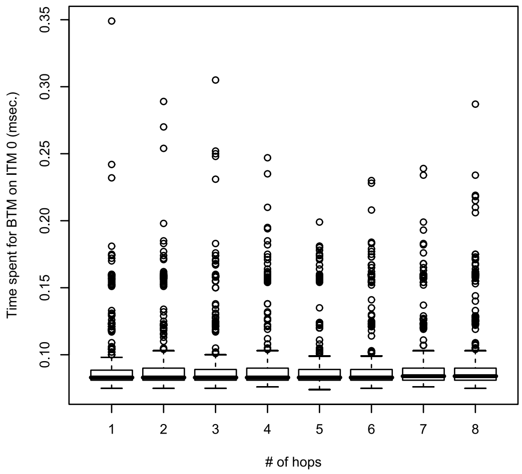



**Fig. 2.17.** The processing time of dummy BTM function

other transit ITMs. The processing time on the dummy BTM function was less than 400 microseconds on each ITM. Fig. 2.17 shows an example of the distribution of the dummy BTM function on the ITM 0. According Figs. 2.13 to Fig. 2.17, the formula described in Section 2.10.1 is adequate for expressing the response time of an traceback query.

In this evaluation, the RTT of an ITM trace request message with an actual implementation of hash-digest-logging method could not be evaluated because of the limited resource of the testbed environment. As sample data, we measured the response time of the software PAFFI[96], which was run on a VMware Workstation 5.0[262]. We measured PAFFI's response time in 1,000 trials both in the case of *Found* and in the case of *Not Found* in the environment shown in Table 2.4. Table 2.5 shows the results.

According to the Table 2.5, the order of the response time of PAFFI is a thousand times as large as the RTT of our ITM implementation; therefore, the border tracking stage would become the bottleneck point on the trial of the inter-domain traceback. We estimated the RTT of an ITM trace request using the result of experiments and the formula described in Section 2.10.1. In the case where each AS used PAFFI as BTS in

**Table 2.3.** The data status of the box-whisker plots on Fig. 2.17

| hop length | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| max. (msec.) | 0.349 | 0.289 | 0.305 | 0.247 | 0.199 | 0.23 | 0.239 | 0.287 |
| mean (msec.) | 0.087 | 0.087 | 0.088 | 0.888 | 0.087 | 0.087 | 0.088 | 0.088 |
| 90th percentile (msec.) | 0.0980 | 0.103 | 0.100 | 0.103 | 0.099 | 0.099 | 0.103 | 0.103 |
| 75th percentile (msec.) | 0.0885 | 0.090 | 0.089 | 0.090 | 0.089 | 0.089 | 0.090 | 0.090 |
| median (msec.) | 0.0830 | 0.083 | 0.083 | 0.083 | 0.083 | 0.083 | 0.084 | 0.084 |
| 25th percentile (msec.) | 0.0810 | 0.081 | 0.081 | 0.081 | 0.081 | 0.081 | 0.081 | 0.081 |
| min. (msec.) | 0.0750 | 0.075 | 0.075 | 0.076 | 0.074 | 0.075 | 0.076 | 0.075 |
| variance | 0.0002 | 0.0003 | 0.003 | 0.0002 | 0.0002 | 0.0002 | 0.0002 | 0.0003 |

**Table 2.4.** The spec of PAFFI on VMware

| Item | Spec |
|---|---|
| Host PC | PowerEdge 1550 |
| CPU | Pentium III 993 MHz |
| Host OS | Windows XP SP2 |
| Guest OS | Red hat 6.2 |
| Memory | 768 MBytes |
| Network | VMNAT |

**Table 2.5.** Response time of the software version PAFFI on VMware

| | Found case | Not Found case |
|---|---|---|
| Max (sec.) | 3.27 | 3.99 |
| Min (sec.) | 0.58 | 0.19 |
| Mean (sec.) | 1.57 | 1.61 |
| Median (sec.) | 1.58 | 1.59 |
| Var | 0.04 | 0.10 |

the 9 hop length topology, the estimated RTT was about 16 seconds in average. Along with this preliminary evaluation result, it was concluded that the bottle neck point of the traceback trials through InterTrack would be the border tracking stage on each AS.

## 2.11 Comparison among Other Architectures

In this section, we compare InterTrack with other inter-domain traceback architectures. For comparison, we take the architectures which have been discussed in IETF: iTrace-CP[250] as one of the extensions of ICMP traceback message, SPIE[184], eIP/iIP[177], RID[158], RID-MEW[113] and InterTrack. Tables 2.6 and 2.7 show the comparison among each architecture in the qualitative view and Table 2.8 describes that of the quantitative view.

Except for iTrace, each architecture is a hierarchical structure. Each structure is characterized by the message protocols for the inter-domain traceback. On the reverse path of a traceback result, except for eIP/IP and InterTrack, each architecture reveals a topology or IP/MAC address to others. RID, RID-MEW and InterTrack are independent from a specific traceback technique.

iTrace-CP and eIP/iIP need the cooperation among ASes to collect traceback information for the reverse path reconstruction because of the characteristic of the traceback packet method. On other architectures, each AS can collect traceback information by itself only.

Authenticating and authorizing components, SPIE, eIP/iIP, and InterTrack are based on the certificate process of sBGP or soBGP and use IPSec ESP to encrypt a TCP session. RID and RID-MEW are authorized and authenticated by the CAs operated by consortium and employ SOAP or HTTPS for the message encryption. We suppose that InterTrack is comparable or may be superior with others in each item of the Tables 2.6 and 2.7.

On quantitative view (Fig. 2.8), several architectures have reported the expected time to finish

Table 2.6. A comparative table on the qualitative view (1)

| approach | protocol | structure | purpose | reverse path | disclosed information |
|---|---|---|---|---|---|
| iTrace-CP | ICMP traceback message | flat | layer 3 traceback of traffic | router hops | IP, MAC, topology |
| SPIE | IP Packet Traceback Protocol | single level hierarchy | layer 3 traceback of a packet | agent (router) hops | IP, topology |
| eIP/iIP | ITM Protocol | two level hierarchy | layer 3 traceback of traffic | AS hops (eIP) router hops (iIP) | AS number, AS path |
| RID | IODEF RID extension | consortium peering | layer 3 traceback, Incident report | AS hops | AS number, AS path, IP |
| RID-MEW | RID MEW extension | three level hierarchy | layer 3 traceback, incident report | AS hops | AS number, AS path, IP |
| InterTrack | ITM trace messages | two level hierarchy | layer 2/3/7 traceback | AS hops | AS number, AS path |

Table 2.7. A comparative table on the qualitative view (2)

| approach | dependent technique | cooperation with others to collect evidences | traceability of messages | authentication | based certification |
|---|---|---|---|---|---|
| iTrace-CP | traceback packet | needed | router ID | — | — |
| SPIE | hash-digest-logging | not needed | message ID, signatures | IPSec | sBGP |
| eIP/iIP | traceback packet | needed | HMAC, signatures | HMAC, IPSec | sBGP, soBGP |
| RID | independent | not needed | message ID, signatures | SOAP/HTTPS | consortium |
| RID-MEW | independent | — | message ID, signatures | SOAP/HTTPS | — |
| InterTrack | independent | not needed | message ID, ITM path list | IPSec | sBGP, soBGP |

Table 2.8. A comparative table on the quantitative view

| | iTrace-CP | eIP/iIP | RID-MEW | InterTrack |
|---|---|---|---|---|
| experiment | ns2 | emulation with implementation | emulation with dummy function | emulation with dummy function |
| average time spent to trace | 18.3 sec. | 38 sec. | 1.6 sec. (test) | 0.25 sec. (test) 16 sec. (estimated) |
| hop length | 20 (router) | 4 (AS) | 7 (AS) | 9 (AS) |
| trees | single tree | 9 trees | single tree | single tree |
| CPU | — — | Pentium III 800 MHz | Pentium IV 3.0 GHz | Pentium III 1.4 GHz |
| tracing | ns2 | actual implementation | dummy (fixed value) | dummy (flexible value) |
| tracing time on each AS | — | — | 0.2 sec. (mid. hop) 0.4 sec. (end hop) | 0.083 usec. (test) 1.6 sec. (PAFFI) |

a traceback trial. Although each experimental environment is different from each other, the average time to spend on a traceback trial was less 40 seconds on iTrace-CP, eIP/iIP, RID-MEW and InterTrack. Even when the sample data on the response time of PAFFI is considered, the average time of a traceback trial on InterTrack is comparable to others.

Through these comparisons, InterTrack can be considered a competitive traceback architecture in practical use.

### 2.12 Summary

Developing an automated inter-domain traceback architecture has long been viewed as impractical due to the barriers on the operational boundaries and the dependence on specific tracking techniques. InterTrack's key contribution is to achieve the inter-domain traceback in a practical way. Because of the phased-tracking apporach of InterTrack, each AS can control a traceback operation on its operation domain by itself only, and can track back an attack on its operation domain with different traceback techniques regardless of traceback systems on other ASes. The federation of internal traceback trials on InterTrack enables ASes to cooperate with others on an inter-domain traceback trial automatically, while concealing the sensitive information of each AS.

Through preliminary experiments, the time spent for an inter-domain traceback in 9 AS hop length was estimated to be 16 seconds in the case where each AS uses hash-digest-logging method. Through discussions and comparisons among other traceback architectures, we explained InterTrack as the competitive traceback architecture against attacks.

In future work, in order to confirm the feasibility of InterTrack, more complex cases where ASes employ various traceback techniques such as specialized routing methods[39, 74, 82, 241], flow sampling methods[40, 189], hash-digest-logging methods[96, 226] etc., will be evaluated.

## 第3章　IP トレースバック相互接続におけるパケットの秘匿性に関する一考察

本章では2章で述べた IP トレースバック相互接続アーキテクチャである InterTrack の実運用において懸念させるパケットの秘匿性についての考察を述べる。また、本章の内容は第2回情報通信システムセキュリティ時限研究会で発表を行った論文を再考し、加筆したものである。

### 内容梗概

IP トレースバック技術の実用化にあたっては、通信の秘密を確保しつつトレースバック方式に対する制約条件を最小化することが求められる。本報告書ではこのような条件のもとでトレースバック方式を相互接続するためのパケット秘匿化方式について、暗号理論からの知見をふまえて考察する。この結果、通信の秘密を保ったまま相互接続可能なトレースバック方式はある程度限られることがわかった。また、悪意ある隣接ドメインを前提とした場合においても秘匿性を維持できることがわかった。

### 3.1 はじめに

不正アクセスに対処するためには不正な IP パケットの発信源を突き止めることが必要となるが、その手がかりとなる IP パケットの始点アドレスが偽造された場合、発信源を特定することは困難である。この問題に対処するための技術として、IP トレースバックの実用化がすすめられている。IP トレースバックについては、これまでに数多くの方式が提案されている。ここではそれらの方式を総称して IP トレースバック方式と呼ぶ。

IP トレースバックの各方式では特定のアルゴリズムを用いているため、迂回攻撃が考案された場合に網全体で IP トレースバック方式を変更する必要がある。またトレースバック方式の秘匿性、時間軸の遡及範囲、精度、対処可能な脅威といった能力はコストとのトレードオフの関係にあることから、各ドメインにたいし費用と安全性に関する選択肢を提供することが望ましい。

このため我々の研究グループでは、各ドメインに

おいて独立した IP トレースバック方式を用いることとし、ドメイン間においては IP トレースバック方式の相互接続を行うアーキテクチャを提案している [84]。また他の研究グループからも類似した提案が提出されている [114, 158] ことから、トレースバック相互接続において現実のネットワーク運用に沿いドメイン化を行うことについては合意形成がなされていると考えてよい。

しかしながら、任意のトレースバック方式の間で相互接続を行うためには平文のパケットの一部が必要となり、通信の秘密を侵害してしまうおそれがある。本報告書では通信の秘密を確保しつつ IP トレースバック方式に対する制約条件を最小化することを目的とし、トレースバック方式を相互接続する上で必要なパケット秘匿化方式について、暗号理論からの知見をふまえて考察する。

本報告書の構成は次のようになっている。3.2 節では既存の IP トレースバック方式を簡単に紹介し、このうち実用化が見込めるものについて、秘匿性要件とその実現可能性について考察する。3.3 節では IP トレースバック相互接続の概要について述べたあと、秘匿性にたいする脅威モデルを構築し、秘匿性の実現方法について考察する。また筆者らが提案する InterTrack[84] モデルにおけるパケットの秘匿化手法について考察する。最後に 3.4 節にてまとめを行う。

## 3.2 IP トレースバック

ここでは IP トレースバック技術の分類法について簡単に紹介する。さまざまな IP トレースバック方式が提案されているが、このように分類をおこなうと入力、出力、制御方法が大きく異なることが分かる。

- **探知情報：**パス特定型は、発信源への通信経路を明らかにする。ソース特定型は、経路ではなく、発信源を明らかにする。
- **追跡対象：**フロー追跡型は、一定の流量をもつフローを対象として逆探知を行う。パケット追跡型は、単一パケットを対象として逆探知をおこなう。パケット標本型は、ある規則にしたがって標本化されたパケットの一部を対象とする。
- **制御方法：**対象ネットワーク内で逆探知パケットを送るインバンド型と、対象ネットワークとは別の制御ネットワークを用いるアウトオブバンド型がある。
- **追跡対象レイヤ：**パケットのどのレイヤを対象とするかによって、レイヤ 2、レイヤ 3、レイヤ 7 と区別することができる。

### 3.2.1 IP トレースバック方式

DoS、DDoS 攻撃に特化したトレースバック方式として、リンク検査方式、およびブラックホール方式 [241, 254] が知られている。これらの方式はフロー追跡型・ソース特定型である。リンク検査方式では被害者の最寄りのルータから DoS の上流となるリンクを特定し、隣接ルータへと順にたどっていくことで発信源を特定することができる。またブラックホール方式では被害者あてのトラフィックのみを特別な経路に誘導することで攻撃パケットが流入しているエッジルータを特定することができる。これらの方式は単一ドメイン内での運用を想定したものであり、すでに実用化が進んでいる。

ダイジェスト方式 [225] はパケットの先頭部分から計算した複数のハッシュ値を Bloom フィルタ [20] に記録することで、IP ヘッダを流用あるいは拡張することなく逆探知を可能にしている。ダイジェスト方式はパケット追跡型・パス特定型であり、レイヤ 3 およびレイヤ 2 を対象とすることができる。

軌跡標本化（Trajectory Sampling）は各ルータが独立して標本化を行うのではなく、経路上のルータで同一パケットを標本化することによりパケットの到達経路を標本化する方式である [60]。標本化対象となるパケットのハッシュ値を指定することで、逆探知が可能となる。本方式はパケット標本型・パス特定型である。本方式はレイヤ 3 だけでなくレイヤ 2 にも適用可能だと考えられる。

特徴量方式はフローの特徴量に注目して IP トレースバックを行うものである。用いる特徴量抽出アルゴリズムとしてはウェーブレット、エントロピ、サポートベクタマシン、時系列解析などが考えられる。本方式はレイヤ 3 だけでなくレイヤ 7 に対しても拡張可能だと考えられる。本方式はフロー追跡型・パス特定型である。

なお、追跡パケット方式 [17] やマーキング方式 [187, 211, 231] などの IP トレースバック方式については、方式研究自体は盛んに行われてきたが、ルータの置き換えを前提としており導入負荷が高いことや、IETF での標準化活動が停止していることなどから実用化の見込みは低いと考える。

### 3.2.2 IP トレースバック方式における秘匿性

本節では IP トレースバック方式における秘匿性要件、なかでも特にドメイン内における秘匿性要件について述べる。現在提案されている IP トレースバック方式の中でパケットの秘匿化が行えるものはダイジェスト方式、特徴量方式、軌跡標本化方式の 3 つの方式である。リンク検査方式は DoS パケットを判別するフィルタをルータに設定する必要があり、このためにアドレスあるいはポート番号などの情報が必要である。そのため、これらの情報を秘匿したまま動作させることはできない。ブラックホール方式についても同様である。

ドメイン内におけるトレースバック方式としてはハッシュ関数を用いたものが多いため、一般論として、ドメイン内における秘匿性を維持するためにはハッシュ関数の一方向性が重要となると考えられる。

なお、秘匿性は電気通信事業者法における通信の秘密により要請されるものであるので、法的要件に対する技術的解決であるということを考慮し、秘匿性の程度を定めることが必要である。ハッシュ関数の一方向性は連続量であり、敵対者が無限の計算資源を持つ場合には総当たり攻撃による原情報の復元も可能であると考えられることから、システムが完全な秘匿性を持つと仮定して法的要件を構築するのは誤りである。

また通信の秘密は、緊急対応など他の法的要請との優先順位の中でのみ語られるということに注意すべきである。具体的には、明らかな攻撃パケットであればその内容を秘匿化し保護する必要はなく、緊急対応を優先し、内容を開示してでも逆探知を行うことが可能である。また、このような緊急対応は機械的な攻撃検知（侵入検知システムなど）によって開始できるという法的見解にもとづき、侵入検知システムと IP トレースバックシステムを連動させることを前提とした方式設計を行うべきである。

一方で、秘匿性については通信の秘密だけでなく攻撃手法の情報拡散防止という観点からも考慮がなされるべきである。つまり、IP トレースバックの問い合せパケットから攻撃手法が獲得できてはならない。緊急対応であっても、限定的な情報開示で済むようなトレースバック方式設計が望まれる。

#### 3.2.2.1 軌跡標本化における秘匿性

軌跡標本化（Trajectory Sampling）は各ルータが独立して標本化を行うのではなく、経路上のルータで同一パケットを標本化することによりパケットの到達経路を標本化しようというものである。具体的には、パケットの不変部分 $P_k$ に対し標本化ハッシュ関数 $H_s$ を適用し、ハッシュ値 $S = H_s(P_k)$ がハッシュ範囲 $R$ に収まっていれば標本化を行う。このとき $P_k$ に対し識別ハッシュ関数 $H_i$ によりパケット ID $H_i(P_k)$ を計算し、これを観測地点に送る。ここで、$R$ として追跡したいパケットのハッシュ値を指定することで IP トレースバックが可能となる。

本方式は近年 IETF PSAMP ワーキンググループにおいて標準化がすすめられていることから有用性が期待できる。また PSAMP 草案は Cisco NetFlow version 9[40] を元にしているため装置実装が容易であり、この観点からも重要性の高い方式である。

しかしながら PSAMP 草案ではハッシュ関数として BOB、IPSX[286] という安全性評価のなされていないアルゴリズムを採用しており、衝突困難性が破られた場合、迂回攻撃につながる危険性がある。また、仮に一方向性が破られた場合、パケット ID から $P_k$ が復元できることになる。文献 [153] では軌跡標本化方式の実装を用いて統計的なハッシュ衝突率や計算処理時間の解析を行っている。しかしながら、文献 [153] では総当りによるハッシュ衝突にかかる計算処理時間などの検証は行っていない。

IPSX のハッシュ値は 16 ビット、BOB では 32 ビットであることから、一般的な暗号学的見解として、衝突困難性は確保できないという。このため迂回攻撃は容易だと結論づけることができる。この点については PSAMP ワーキンググループに対し指摘をおこなっているところである。

PSAMP 草案では、観測地点はパケット ID だけでなく、$P_k$ のほぼ任意のフィールドを標本として送るよう指定することができるため、秘匿性を確保するためにはハッシュ関数の安全性評価と並んで運用現場におけるセキュリティ管理が重要となる。

#### 3.2.2.2 ダイジェスト方式における秘匿性

ダイジェスト方式では MD5 や SHA1 などの一方向性ハッシュ関数を用いてネットワーク中を転送されるパケットのハッシュ値をデータベースに記録し、追跡時にハッシュ値がデータベース上に存在するか否かで追跡を行う。

ダイジェスト方式では Bloom フィルタを用いるた

め、偽陽性を抑制するためにハッシュ関数を複数回計算することになる。ここで単一のハッシュ関数を用いて $H_i(P_k) = H(i||P_k)$（$0 \leq i \leq n$）として $n$ 個のハッシュ値を計算した場合、一方向性がある程度、劣化することになる。しかしながら多くの文献では、一方向性が必要とされない状況において、このような複数ハッシュ値の計算方法が示唆されている [18]。

一方向性が劣化せず、かつソフトウェア実装でさらに高い性能を得ることができる計算方法は単純である。単一ハッシュ値を計算し、数十ビット単位でワード分割をおこない、複数のハッシュ値とする方法がこれまでに提案されている [54]。なお、同文献で提案されているハッシュ値の高速計算手法（double hashing）は安全性が検証されておらず、用いるべきでない。

ダイジェスト方式の実装である SPIE[15] では、秘匿性について実装上の問題点がある。問い合わせ時に追跡要求メッセージに付与するパケット情報は、各記録エージェントごとや時間ごとに用いているハッシュ関数が異なるため、パケットそのものが必要となり、パケット情報の秘匿化が行えない。文献 [226] や文献 [242] では Bloom Filter の記録と追跡要求メッセージに付与する必要のあるパケット情報を IPv4 パケット、IPv6 パケットそれぞれの場合について計測結果を元に考察し、IPv4 パケットの場合は IP ヘッダとペイロード 8 バイト部分（TCP/UDP ヘッダ部分）、IPv6 パケットの場合は IP ヘッダと拡張ヘッダで十分識別できるとしている。SPIE では追跡パケットが秘匿化されないかわりに、トレースバック網以外へのパケット情報の漏洩を防ぐためにマネージャ間、記録エージェント間の通信を IPSec ESP で暗号化して通信を行っている。

#### 3.2.2.3 特徴量方式における秘匿性

単一パケットのハッシュ値ではなく、フローの特徴量に注目して IP トレースバックを行うことも考えられる。例えば各リンクにおいて流量を対象としてウェーブレット解析をおこない、ウェーブレット係数の相関をもとに異常フローの経路を逆探知できる可能性があることが複数の文献で示唆されている [123, 135]。また各リンクにおいてアドレス、ポート等からエントロピを計算し、多次元空間におけるクラスタリングを行うことによって流量に依存せずに異常フローを検知できることも指摘されている [129]。

information harvesting
- (a) deliberate exposure
- (b) eavesdropping attack
- (c) spoofing attack
- (d) redirection attack
- (e) traceback request forgery
- (f) false reply attack

**図 3.1.**　IP トレースバック相互接続における秘匿性に対する攻撃木

これらの特徴量を利用したトレースバック方式では、単一パケットに基づいて問い合わせを行う方式と比較して秘匿性が高い。しかしながら、異なる特徴量抽出アルゴリズムにもとづくトレースバック方式の接合は不可能であるため、少なくともドメイン境界においてはパケットの標本化を行うなどして接合可能なトレースバック方式との相互接続性を保つ必要がある。

また特徴量方式の課題として、オンライン解析の実現例が少なく、多くがオフライン解析であることと、著しい特徴が現れない場合は追跡が困難であることが挙げられる。

## 3.3 IP トレースバック方式の相互接続

### 3.3.1 IP トレースバック相互接続における秘匿性問題

ここではドメイン間の秘匿性要件について述べる。ドメイン間では、利益相反する他事業者に対してもトレースバック問い合せパケットが送られるため、攻撃判定後のトレースバックであってもパケットを秘匿化することが望ましい。

ドメイン間でのパケット秘匿化手法としては、ハッシュ関数、群暗号、プロトコルによる解決などが考えられる。これらについては次節で考察する。

何らかの解析能力をもつ敵対者が隣接するドメインにおいて存在した場合、さまざまな攻撃が考えられる。ここでは IP トレースバック相互接続における攻撃木のうち、秘匿性に関係あるものについて述べる。図 3.1 に攻撃木を示す。(a) はノードの乗っ取りによる情報漏洩である。バッファ・オーバーランは近年のコンパイラ技術の進展によって防止できるため [289]、実際には脆弱なパスワードによる権限奪取が問題となると考えられる。このため敵対者が隣接ドメインに存在すると考えるのが現実的である。

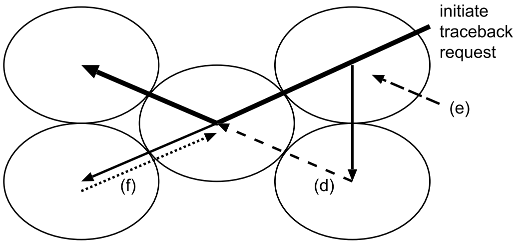



**図 3.2.** ドメイン間における情報収集攻撃の類型

(c) は相互接続先へのなりすましである。これらの攻撃は一般的なもので、IP トレースバック相互接続に固有の対策手法を考えうるものではない。これらの攻撃は起きうるものとして相互接続システムを設計し、一般的な対策を講じるのが現実的である。

いっぽう、攻撃 (d)–(f) はトレースバック相互接続に固有の攻撃である。これらについては、図 3.2 を併用し説明する。図中の実線はトレースバックの追跡要求を表す。このうち、太線で示した部分が追跡結果である。(d) は隣接ドメインから来た追跡要求を他へ転送することによる情報収集である。これは、攻撃パケットが実際には通過していない隣接ドメインにたいしても追跡要求を送ってしまった場合に起きうる。この結果として攻撃経路に関する情報を当事者以外に漏洩してしまう可能性がある。(e) は追跡要求を捏造することによる情報収集である。ランダムな値をもつ追跡要求を送ることによって、隣接する組織がどのような相手と通信しているかを調べるなどの意図が考えられる。(f) は攻撃パケットが実際には通っていないにもかかわらず、通った旨の偽りの追跡応答を返すことによる情報収集である。これによってトレースバック精度の劣化をはかるといった意図が考えうる。

(d)–(f) への対策としてはまず、追跡要求における情報量を減らし、収集できる情報量を減らす必要がある。ここで、追跡対象となるパケットが通過したドメインでは原情報を観測しており、通過しなかったドメインのみが敵対者として考えうるということに注目する（通過したドメインに敵対者がいた場合、(a)(b) によって原情報を復元できるため (d)–(f) を考慮する必要がない）。パケットが通過しなかった可能性があるドメインに追跡要求を送るのは追跡の初期段階のみとし、詳細な追跡情報を得るための条件として、初期段階の追跡応答において原情報の観測証明を求めることで (d)–(f) を解決できる。詳細については 3.3.2.3 節で述べる。

### 3.3.2 ドメイン間でのパケット秘匿化方式の検討

#### 3.3.2.1 群暗号

文献 [18] では群暗号（group cipher）を用いることで、検索内容を明かすことなく Bloom フィルタから検索を行う手法を提案している。この方法をドメイン間での追跡要求に用いることもできるが、信頼できる第三者が必要であり、かつ隣接ドメインとの変換鍵 $r_{A,B}$ を信頼できる第三者において保持する必要がある。また唯一の群暗号として知られる Pohlig-Hellman 暗号はべき乗と素数での剰余演算からなり、計算時間が問題となる。

#### 3.3.2.2 ハッシュ関数

計算量やハードウェア実装の容易さ、秘匿性要件を考慮すると、相互接続においてもハッシュ関数を用いるのが最も現実的だと考えられる。ただし、ハッシュ関数が危殆化した場合に備えて、IKE（Internet Key Exchange）[118] のように使用アルゴリズムに関する合意手順を隣接ドメインとの間で用意する必要がある。しかしながら IKE とは異なり、隣接ドメイン間だけでなく経路上のすべてのドメインにおいて使用アルゴリズムを揃える必要があるため、合意手順についても何らかの工夫が必要である。例えば全ドメインの時刻同期を前提とし、半年に一度、使用アルゴリズムの合意手順を動作させるといったことが考えられる。

なおトレースバック相互接続ではオフライン攻撃（pre-computation attack）が起きないので、ハッシュ関数に求められる性能は高くない。このため SHA-1 などの衝突困難性の劣化が懸念されるハッシュ関数を用いたとしても、実時間で衝突を起こす方法が考案されない限り問題とはならない。このため、我々の研究グループを含め 8 社で構成するトレースバック研究ユニットでは SHA-1 アルゴリズムを用いる方針である。

NIST や IETF ではハッシュ関数の衝突困難性の劣化をふまえて、よりビット数の大きな SHA-256 へ移行するよう勧告が出されている [162]。SHA-256 IP コアも提供されており、またネットワークプロセッサへの実装も始まっていることから、トレースバック相互接続の製品化においては SHA-256 を出発点として用いるのが適当だと考えられる。

#### 3.3.2.3 段階的開示

ここでは、ドメイン間の追跡要求においてハッシュ関数を使った場合において、3.3.1 節で述べた攻撃木における (d)–(f) にどのように対処すればよいかについて考察する。

まず原情報であるパケットに対し、パケット ID $H(P_k)$ を求める。ここで $H$ は適当な SHA とする。これを適当な長さの $k_i$ と $k_r$ に分割する。イニシエータ $I$ とレスポンダ $R$ はそれぞれ $P_k$ を観測しているので、$k_i$、$k_r$ を独立して計算できる。初期段階の追跡要求では $k_i$ のみを送る。

$$I \rightarrow R\colon k_i$$

応答において $k_r$ で暗号化を行う。ここで $x$ は乱数である。レスポンダは $P_k$ を観測しなかった場合においても、$k_r$ を乱数から生成し暗号化を行う。

$$R \rightarrow I\colon \{x, k_i, R\}_{k_r}$$

$k_r$ で復号化でき、$(x, k_i, R)$ が得られれば偽りの追跡応答でないことが分かる。これにより (f) が解決できる。また $(x, k_i, R)$ が得られなかった場合は追跡が失敗したことがわかる。いずれの場合も、$x$（または適当な乱数）をレスポンダに送り返す。

(d) では $k_r$ を知らないため復号化できない。また追跡の成否にかかわらず同一のやりとりが繰り返されるため、中間者において $I$ と $R$ の間で通信があったかどうかを知る術はない。

さらに (e) つまり追跡要求の捏造を行った場合、$\{x, k_i, R\}_{k_r}$ が復号できないため $x$ を送り返すことができず、露呈する。

なお文献 [308] では中間者攻撃を防ぐために AS 番号をハッシュ化して埋め込む方法を提案しているが、この方法ではパケットを観測していない AS がトレースバック相互接続において問い合わせを中継することができない。トレースバック相互接続システムがインターネットの一部でのみ採用されている場合、部分的な被覆をもつシステム間をつなぐ善意の中間者と、悪意ある中間者を区別することはできないという点に留意する必要がある。

### 3.3.3 InterTrack モデルにおける秘匿性

我々の研究グループが提案し開発している InterTrack では追跡要求の処理を追跡要求受付、AS 境界、AS 間、AS 内の 4 段階に分けた追跡モデルを採用している。AS 間追跡において、3.3.2.3 節で述べたような段階的開示プロトコルを採用することで秘匿性を高めることができると考える。

InterTrack モデルでは AS 間の問い合せに用いるハッシュ関数 $H$、AS 境界への問い合せに用いる関数 $Q_b$、AS 内への問い合せに用いる関数 $Q_d$ を分けることができる。ここで、何らかの縮約関数 $R_b$、$R_d$ を用いることで以下の関係が成り立つのであれば問い合せを接合することができる。

$$R_b(H(p)) = Q_b(p)$$
$$R_d(H(p)) = Q_d(p)$$

ここで $p$ はパケットのヘッダないしペイロードの一部である。仮に、ハッシュ関数と問い合せ関数が逆の関係、つまり $H(p) = R_b(Q_b(p))$ であった場合は隣接ドメインの問い合せからドメイン内の問い合せに変換できないため接合に失敗する。

関数 $Q_b$、$Q_d$ を分けているのは次のような理由による。AS 境界は同一ドメイン内のドメイン境界に対する問い合せであるため、例えばハッシュ関数であればビット数を多くとり、偽陽性を低くするのが妥当だと考えられる。また AS 内に対する問い合わせは、事業者が収容する複数の顧客側設備に対して送られる可能性があるため、情報量が少ないほうがよい。

## 3.4 まとめ

本報告書における議論をふまえての結論として、以下のことが言える。トレースバック方式の相互接続において秘匿性を確保するためには、パケットヘッダの不変部分をもとにハッシュ値を計算する方式を一つ標準化する必要があり、またハッシュ関数の経年劣化を前提とした合意手順を確立する必要がある。さらに、各ドメインでは少なくとも一か所において、ドメイン間の追跡要求に含まれるハッシュ値から AS 境界、AS 内の問い合せに変換する機能を持つことでドメイン内とドメイン間のトレースバック機能の接合を行うことができる。また、このような接合方法によって相互接続できるトレースバック方式は、ダイジェスト方式、特徴量方式、軌跡標本化方式である。

## 謝辞

本研究の一部は科学技術振興調整費重要課題解決型研究「セキュリティ情報の分析と共有システムの

開発」の成果を反映している。また本報告書は情報通信研究機構委託研究テーマ「インターネットにおけるトレースバック技術に関する研究開発」における議論をうけて論点整理を行ったものである。

本報告書の執筆にあたり、神戸大学の森井昌克先生、「セキュリティ情報の分析と共有システムの開発」の共同研究者の皆様、情報通信研究機構委託研究事業のトレースバック研究ユニットの皆様には貴重なご示唆をいただきました。ここに感謝申し上げます。

## 第4章　おわりに

2006 年度の Traceback ワーキンググループの活動は IP トレースバック相互接続アーキテクチャの設計と実装、および実運用におけるパケットの秘匿性に関する考察を行った。現状としては、DNS リフレクションやボットネットなどの踏み台攻撃への対応、パケットの秘匿性の確保や高速広帯域ネットワークへの対応、方式のコストパフォーマンスなど IP トレースバックの実用化に向けてはまだまだ研究として取り組むべき課題が残されている。

2007 年度の活動予定としては、踏み台攻撃への対応、秘匿性の確保などの基礎研究をおこないつつ、InterTrack の実装を用いた IP トレースバックプローブを WIDE バックボーンに設置し、WIDE バックボーン内に IP トレースバックテストベッドを構築し、実ネットワークを用いた IP トレースバックおよび IP トレースバック相互接続アーキテクチャの実験を行う予定である。