A End-to-end Source Address Validation Solution for IPv6
draft-wu-sava-solution-e2e-ipv6-00.txt

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on August 11, 2007.

Copyright Notice

Copyright (C) The IETF Trust (2007).
Abstract

This document describes the detail mechanism and protocol definition of a light-weight signature based and end-to-end deployed method for preventing the spoofing of source IPv6 address.

Table of Contents

1. Requirements Language .................................. 4
2. Introduction ............................................ 5
3. Terminology ........................................... 6
4. Overview of the Solution ................................. 7
   4.1. Basic Mechanism .................................... 7
   4.2. System Architecture ................................ 7
   4.3. Advantages ......................................... 9
      4.3.1. Deployment Incentive ......................... 9
      4.3.2. Incremental Deployment ..................... 9
5. Data Plane .............................................. 10
   5.1. Packet Format ...................................... 10
   5.2. Modification to the Routers ....................... 11
   5.3. Presentation of Prefix Ownership ................. 12
   5.4. MTU Issues ........................................ 12
   5.5. Packet Processing Steps ......................... 13
      5.5.1. Processing Steps for Incoming Packets .... 14
      5.5.2. Processing Steps for Out-going Packets ... 14
6. Control Plane ........................................... 16
   6.1. Overview ........................................... 16
   6.2. Security Issues .................................... 16
   6.3. Control Data in Registration Server .............. 16
   6.4. Control Data in AS Control Server ............... 16
      6.4.1. E2E Alliance Member List .................. 17
      6.4.2. Local Prefix List ............................ 17
      6.4.3. Prefix-AS Mapping Table .................... 17
      6.4.4. AS-In Signature Table ....................... 18
      6.4.5. AS-Out Signature Table ...................... 18
7. Protocol Between Registration Server and AS Control Server . 19
   7.1. Overview ........................................... 19
   7.2. Message Formats ................................... 19
      7.2.1. INFO_REQUEST Message Formats ............. 19
      7.2.2. INFO_REQ_ACK Message Formats .............. 20
      7.2.3. INFO_REQ_NAK Message Formats .............. 21
1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].
2. Introduction

The Internet is a decentralized system which basically provides best effort, packet-based data forwarding service. In the forwarding process, the device (router, switch, gateway, etc.) forwards IP packets based on the destination IP address. In most cases, the source IP address is not checked. The lack of source IP address checking makes it easy for the attackers to spoof the source address. And it has facilitated many existing attacks in the Internet.

Some mechanisms, such as Ingress Filtering [RFC2827], iTrace [I-D.bellovin-itrace], SAVE [Li02], etc., has been proposed to solve the problem of source IP address spoofing. However, these mechanisms are not widely deployed in the Internet. There are two main reasons that impede the deployment: the lack of enough incentive for deployment, and the lack of capability for incremental deployment.

In [Bre05] Bremler-Barr and Levy proposed a idea to prevent source address spoofing. A unique temporary key is shared between two ASes. When a packet is leaving a source network, it is tagged with the key. When this packet arrives at the destination network, the key is verified and removed. This mechanism shows its benefits both at providing incentive for deployment and at ease of incremental deployment. However, the detail mechanism is not designed and some issues were not discussed, e.g., MTU issue, the presentation of address prefix ownership, etc.

This document makes detail specification for a light-weight signature based end-to-end deployed source address validation solution for IPv6. In addition to this document, we also finished the prototype implementation, and verified the mechanism in CERNET2 native IPv6 environment (7 core nodes and 10 ASes had been deployed).

Currently we only consider how to apply the mechanism in the IPv6 network, because the deployment is more feasible for the next generation Internet. For applying it in the IPv4 network, the mechanism is quite similar.

The section 4 provides an overview of the solution. The section 5 describes the mechanisms at data plane. The section 6 provides an overview for the control plane. The section 7 describes protocol between registration server and AS control server. The section 8 describes protocol between two AS control servers.
3. Terminology

E2E Alliance: A group of ASes that apply E2E mechanism to prevent source address spoofing.

Registration Server (REG): A server that maintains the E2E Alliance member list.

AS Control Server (ASC): A server that represents one AS member to communicate with Registration Server and other AS members in E2E Alliance.

AS Edge Router (AER): A router deployed at the edge of AS that processes packets with E2E mechanism.

Signature: The 48-bit string carried on each data packet when the packet is transmitted between two AS members in the E2E Alliance.

E2E Option: Option in IPv6 Hop-by-Hops Options header that carries 48-bit signature required by E2E mechanism.

E2E Header: A type of IPv6 extension header that carries 48-bit signature required by E2E mechanism.
4. Overview of the Solution

4.1. Basic Mechanism

The basic ideas of E2E mechanism are as follows:

- All ASes that applying E2E mechanism form an E2E Alliance.
- Each AS in the E2E Alliance guarantees the validity of the source IP address of the packet generated by this AS.
- When a packet is leaving its own original AS, if the destination IP address belongs to some AS member in the E2E Alliance, the edge routers of this AS looks up for the signature based on the destination AS number, and tags a signature to the packet.
- When a packet is arriving at the destination AS, if the source address of the packet belongs to some AS member in the E2E Alliance, the edge routers of the destination AS looks up for the signature based on the source AS number, then the signature carried in the packet is verified and removed.

There are several very important assumptions for the feasibility of E2E mechanism:

- It is hard for an attacker to sniff the backbone between AS in the Alliance.
- Even some transit ASes don’t join the E2E Alliance, they will not try to be harmful to the E2E Alliance.

Under these assumptions, we can use shared random number as the signature, instead of using cryptographic method to generate the signature.

4.2. System Architecture

Here we briefly describe the system architecture, as shown in Figure 1.
There are three major components in the system: the Registration Server (REG), the AS Control Server (ASC), and the AS Edge Router (AER).

The Registration Server is the "center" of the E2E Alliance. It maintains a member list for E2E Alliance. It can support two major functions:

- Process the request from the AS Control Server, to get the member list of the E2E Alliance.
- When the member list is changed, notify each AS Control Server.

Each AS deploying the method should have an AS Control Server. The AS Control Server has three major functions:

- Communicate with the Registration Server, to get the up-to-date member list of E2E Alliance.
- Communicate with the AS Control Server in other member AS in E2E Alliance, to exchange the update in prefix ownership, and to exchange the signature.
- Communicate with all edge routers of the local AS, to configure the processing component on the edge routers.

The AS Edge Router does the work of adding signature to the packet at the sending AS, and the work of verifying and removing the signature at the destination AS.
In the design of this system, we try to decrease the burden of Registration Server. Most of the control traffic happens between AS Control Servers.

4.3. Advantages

4.3.1. Deployment Incentive

The E2E mechanism presented in this document provides strong incentive for the network operator to deploy it; second, it can be incrementally deployed.

The incentive provided by the mechanism shows at three aspects:

o The members in the E2E Alliance may apply some different service mechanism for the traffic from E2E Alliance members and non-members. Traffic from other E2E Alliance members can get better service.

o When some abnormal traffic is coming from some other E2E Alliance member, it is easier to trace back the origination.

o If the attacking traffic targeting one E2E Alliance member by spoofing the source IP address of another E2E Alliance member, the AS owning the spoofed address can easily prove its innocence. This can help to decrease the overhead of network management.

4.3.2. Incremental Deployment

E2E mechanism can be incrementally deployed in the Internet, the main reason is that it is an end-to-end solution. It does not rely on the modification to the middle path routers.

E2E mechanism also has some other advantages. It does not rely on route information or path information. So it is not influenced by the dynamics of route changes and route flaps. And it is feasible in the multihoming environment.
5. Data Plane

5.1. Packet Format

We must find some space in the IP packet to carry the signature. The packet header of IPv6 packet is quite simplified. It is hard to find space available. One possible way is to make use of the 8-bit Traffic Class area and the 20-bit Flow Label area [RFC2460]. But since it still can not hold the signature with 28 bits, and this usage may break some other protocols that use these two area, we decide not to carry the signature in the IPv6 Header. Instead, we design a new E2E Option in Hop-by-Hop Options header [RFC2460] to carry the signature.

The signature carried in the E2E Option is 48-bit long. In [Bre05], the authors described a scenario how two hosts can collaborate to guess the signature, and the original design for the signature is 32-bit long. However, in case of thousands of machines are controlled to cooperatively guess the signature, 32-bit signature is vulnerable. So the signature is extended to 48-bit long.

The E2E Option has the following format:

```
+----------+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 0 0 1 | x x x x x | 0 0 0 0 0 1 1 0 | Signature (1) |
+----------+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Signature (2)                           |
+----------+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The first two bits of the first byte are zero. If the processing IPv6 node does not recognize E2E Option, it just skips over this option. The third bit is 1, since the option data may change en-route. The next five bits is the Hop-by-Hop Option Type number, are left to be specified. The second bit indicates the length of the option data. The length of E2E Option data is 6-byte. Next is the 48-bit signature data.

There MUST only be one option of this type, regardless of value, per Hop-by-Hop header.

Another possible way to carry signatur is to insert an extension header. The E2E Header has the following formats:

```
+----------+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Next Header |  Hdr Ext Len |  Signature (1) |
+----------+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Signature (2)                           |
+----------+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

If signature is carried in the Hop-by-Hop Options header, to keep the 8-octet alignment required by RFC2460 [RFC2460], some extra bytes should be inserted for padding. Also considering the possibility that there may be Hop-by-Hop Options header already in the packet, the processing at the edge router is relative complex. On this aspect, carrying signature in the extension header is relatively simpler.

The advantage of using Hop-by-Hop Options header is that it will not cause ICMP packet with value "unrecognized Next Header type encountered". Routers that don’t know E2E Option can just skip this option. While packet with the extension header may cause ICMP packet with value "unrecognized Next Header type encountered", depending on the specific router implementation.

5.2. Modification to the Routers

The AS edge routers should support the following three functions:

- Drop those coming packets whose source address belongs to the local AS. Such packets may be generated from spoofing of source address, or from loop of route. In both cases, the packets should be dropped.

- Tag signature to the out-going packets whose destination address belongs to another AS in the E2E Alliance.

- Verify and remove the signature of the incoming packet. A packet with signature SHOULD NOT reach inside the AS.

To support these functions, some components should be added to the edge routers:

- Local prefix list table. It stores all prefix owned by the local AS. It is used in processing the incoming packet, to check the source address in the packet.

- Prefix-AS mapping table. It maps prefix to the AS which owns the prefix. It is used both in processing the incoming packet and in processing the out-going packet.

- AS-In signature table. It stores the signature for the incoming traffic.

- AS-Out signature table. It stores the signature for the out-going traffic.
5.3. Presentation of Prefix Ownership

Each AS should present the ownership of prefix. Since Longest Prefix Match is applied in the current forwarding lookup method, in the presentation of the prefix ownership we should also notice this property. In a prefix table, there is not only the prefix block that owned by this AS, but also some small prefix block that not owned by this AS.

Thus the prefix table entry is organized as:

- Prefix: prefix address.
- Prefix length: length of the prefix.
- Flag: whether the prefix is owned by this AS. If the prefix is owned by the AS, the flag is YES; if the prefix is not owned by the AS, the flag is NO.

In checking the prefix table to get the ownership of the prefix, there are three cases:

- No matched prefix: the prefix is not owned by the AS.
- The prefix is matched, the flag is YES: the prefix is owned by the AS.
- The prefix is matched, the flag is NO: the prefix is not owned by the AS.

5.4. MTU Issues

Since E2E Option in Hop-by-Hop Options head is inserted into the outgoing packet, MTU issues should be considered.

The length of E2E Option is 8-byte. If there is no Hop-by-Hop Options header in the original packet, 16 bytes will be added to the packet. If there has been Hop-by-Hop Options header in the original packet, 8 to 16 bytes will be added to the packet. For simplifying the problem, we always assume that 16 bytes will be added to the packet.

The problem is depicted in Figure 2. Host A in the AS-1 sends packet to Host B. The MTU for the output link of the edge router is x. The AER sends an ICMP "Packet Too Big" [RFC2463] message to Host A, and Host A changes its MTU to x-16, i.e., minus length of E2E Option. There is a gateway between Host A and Host B, and MTU of one link is y (y < x). The gateway will send an ICMP "Packet Too Big" message to
Host A, and Host A will change its MTU to Host B to \( y \). However, after forwarded by the AER, the packet size becomes \( y+16 \), and it still can not accepted by the gateway.

\[
\begin{array}{c}
\text{mtu} = x-16 \quad \text{mtu} = x \quad \text{mtu} = y < x \\
\text{A} \quad \text{AER} \quad \text{GW} \quad \text{B} \\
\text{mtu} = y \quad \text{y} + 16 \quad \text{y} + 16 > y \\
\text{AS-1}
\end{array}
\]

Figure 2. MTU issue

Someone may ask why there is such problem, if comparing with some tunnel mechanism. The reason is that AER router inserts E2E option but doesn’t change the source address of the packet. Thus the down-stream gateway sends ICMP "Packet Too Big" to the host, not to the AER.

Currently we have two solutions to this problem. Both of the solutions are not so perfect, but they are feasible.

One possible way is to set the MTU at the AER to be 1280, which is the minimum MTU for the IPv6. Thus there should be no ICMP "Packet Too Big" message received from the down-stream gateways. The disadvantage of this solution is that it doesn’t make good use of the available MTU. Also, if the MTU at the AER is 1280, MTU at the host will be 1264. We doubt whether host will accpet such a MTU, since it is smaller than the IPv6 minimum MTU.

The other possible way is to let AER catch all coming ICMP "Packet Too Big" message", and decrease the value in the MTU area by 16 before forwarding it into the AS. The advantage of this solution is that it can make good use of the available MTU. But such processing of ICMP packet at AER may become a destination of DoS attack.

5.5. Packet Processing Steps
5.5.1. Processing Steps for Incoming Packets

For a packet coming into the AS, the edge router will process it in the following steps:

1. Check the destination address with the local prefix list table. If the destination address matches with the prefix of the local AS, go to the next step; else, just forward it.

2. Check the source address with the local prefix list table. If the source address matches with the prefix of the local AS, drop the packet.

3. Check the source address with the prefix-AS mapping table. If the source address belongs to some AS member in the E2E Alliance, go to the next step; else, forward it. Before forwarding the packet, the packet should be checked to see whether it has E2E option. It is possible to have some cases that the sending AS has changed its prefix list but this information has not reached the local AS. If the packet has E2E Option, the edge router should remove the E2E Option before forwarding this packet into the local AS.

4. Verify the signature in the packet with the record in the AS-In signature table. If the signature in the packet is invalid, drop the packet; else forward it.

In the implementation, some method can be applied to simplify the processing steps:

- The first step could be combined to the normal forwarding lookup for a packet. The local prefix can be added to the forwarding information table (FIB), with some special flags.

- The second step and the third step can be combined to one step. The prefix of local AS can be treated as a special case in the prefix-AS mapping table.

5.5.2. Processing Steps for Out-going Packets

For a packet going out of the AS, the edge router will process it in the following steps:

1. Check the source address of the packet with the local prefix list table. If the source address matches with the prefix of the local AS, go to next step; else, forward it.

2. Check the destination address with the prefix-AS mapping table. If the destination address belongs to some AS member in the Alliance,
go to next step; else, forward it.

3. Get the signature from the AS-Out signature table with the destination AS number. Insert the E2E Option into the packet and forward it.

In the implementation, the first step and the second step can be combined to simplify the processing steps. The prefix of local AS can be treated as a special case in the prefix-AS mapping table.
6. Control Plane

6.1. Overview

In the control plane, the major issues are "two systems" and "two protocols". The "two systems" are the Registration Server and the AS Control Server. The "two protocols" are the protocol between Registration Server and AS Control Server, and the protocol between two AS Control Servers.

6.2. Security Issues

How to secure the communication between the servers is a very important issue. To avoid the usage of PKI, we decide to use TCP as the transmission protocol. TCP provides some means of security, it is hard to set up the communication with these servers with spoofed source address.

Since we’ve made the assumption that the link between the ASes is hard to sniff, we don’t worry about the control data communication to be sniffed. One possible attack is to try to "insert" fake data into the data stream between two servers, by guessing the Seq and Ack of TCP packet. But since normally the communication between these servers is short flows, it is hard to make such attack.

6.3. Control Data in Registration Server

In the Registration Server, a list for all the members in the Alliance is maintained. A list entry has following information for the member:

- AS number: 16 bit
- Address of AS Control Server: 128 bit, for IPv6 address.
- Last action: last operation on this AS member, 8 bit. The operation may be Add or Delete.
- Action ID: each action on the member list should have a unique ID number. This number should increase by one for each new action.

The operation on the list may be "add a member" or "delete a member". The IP address of the AS Control Server is negotiated by other means.

6.4. Control Data in AS Control Server

In the AS Control Server, there are five data tables as follows:
6.4.1. E2E Alliance Member List

This information of E2E Alliance list is updated from the registration server. Different from the list on the Registration Server, only AS number and the address of AS Control Server are required for each member entry. For the whole list, one "last action ID" value is maintained. This value is the ID of latest action learned from the Registration Server.

6.4.2. Local Prefix List

The local prefix list maintains a list for the prefix owned by the local AS. The presentation of prefix ownership has been discussed in Section 5.3. For each entry in the prefix list, the information for a prefix includes prefix, prefix length, and flag, just as described in Section 5.3. There are two additional data areas used to help manage the prefix list:

- Last action: last operation on this AS member, 8 bit. The operation may be Add or Delete.
- Action ID: each action on the local prefix list should have a unique ID number. This number should increase by one for each new action.

6.4.3. Prefix-AS Mapping Table

The prefix-AS mapping table maintains the prefix ownership information of all other AS members in the E2E Alliance. For ease of managing the table, the table should support both mapping from prefix to AS number and mapping from AS number to all owned prefix. The details on operation of prefix-AS mapping table will be discussed in Section 8.
6.4.4. AS-In Signature Table

The AS-In table stores the incoming direction signatures with each peer AS members in the E2E Alliance. The details on operation of AS-In signature table will be discussed in Section 8.

6.4.5. AS-Out Signature Table

The AS-In table stores the out-going direction signatures with each peer AS members in the E2E Alliance. The details on operation of AS-Out signature table will be discussed in Section 8.
7. Protocol Between Registration Server and AS Control Server

7.1. Overview

The function of the protocol between Registration Server and AS Control Server is to spread the information of E2E Alliance list to all AS member in the E2E Alliance. We design two mechanisms to achieve the goal. First, the AS Control Server may initiate a request to the Registration Server to get the list information. Second, when some change happens at the Registration Server, it should notify all AS Control Servers. The communicate is based on TCP connections.

7.2. Message Formats

This section describes message formats used in the protocol between Registration Server and AS Control Server. There are four types of messages: INFO_REQUEST, INFO_REQ_ACK, INFO_REQ_NAK, INFO_NOTIF.

7.2.1. INFO_REQUEST Message Formats

After a transport protocol connection is established between Registration Server (REG) and AS Control Server (ASC), ASC sends INFO_REQUEST message to the REG.

The INFO_REQUEST message contains the following fields:

```
| Command | My Autonomous System | Action ID Parameter |
```

Command: This 1-octet unsigned integer indicates the type code of the message. For INFO_REQUEST message, the command value is 1.

My Autonomous System: This 2-octet unsigned integer indicates the Autonomous System number of the sender.

Action ID Parameter: This 4-octet unsigned integer indicates the action ID parameter for this request. It’s the latest action ID that ASC getting from the REG. If the ASC wants to get the information of all the E2E Alliance member, the value should be set to 0.
7.2.2. INFO_REQ_ACK Message Formats

After REG receives INFO_REQUEST message from ASC, if the request is valid, the REG will send back INFO_REQ_ACK message to the ASC.

The INFO_REQ_ACK message contains the following fields:

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

Command: This 1-octet unsigned integer indicates the type code of the message. For INFO_REQ_ACK message, the command value is 2.

Number of Data Records: This 4-octet unsigned integer indicates the number of the total records returned.

Last Action ID: This 4-octet unsigned integer indicates the latest action ID of the records returned.

Data Records: This is a variable length field that contains a list of the Alliance member information. Each AS member info record contains the following field:

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

Autonomous System Number: This 2-octet unsigned integer indicates the Autonomous System number of the member.

Server Address: This 16-octet data indicates the IPv6 address of the AS Control Server for this AS member.
Last Action: This 1-octet unsigned integer indicates the last action for this AS member. The value of this field may be:

- Add(1): last action is "add" for this AS member.
- Delete(0): last action is "delete" for this AS member.

### 7.2.3. INFO_REQ_NAK Message Formats

After REG receives INFO_REQUEST message from ASC, if the request is invalid, the REG will send back INFO_REQ_NAK message to the ASC.

The INFO_REQ_NAK message contains the following fields:

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Command    |  Error Code   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- **Command**: This 1-octet unsigned integer indicates the type code of the message. For INFO_REQ_NAK message, the command value is 3.
- **Error Code**: This 1-octet unsigned integer indicates the error type in processing this request. The value of the error code may be:
  - 1: invalid AS number. The AS that sends the request to the REG has not joined the E2E Alliance.
  - 2: invalid ASC address. The address of the sending host does not match the address in the E2E Alliance member list.

### 7.2.4. INFO_NOTIF Message Formats

If there are some update to the E2E Alliance member list at the REG, REG should send INFO_NOTIF message to ASC of all the AS member in the E2E Alliance.

The INFO_NOTIF message contains the following fields:

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Command    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Latest Action ID                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
Command: This 1-octet unsigned integer indicates the type code of the message. For INFO_NOTIF message, the command value is 4.

Latest Action ID: This 4-octet unsigned integer indicates the action ID of lastest update to the E2E Alliance member list at the REG.

7.3. Possible Scenarios

7.3.1. PASC Initiate Request to the REG

ASC may initiate sending request to the REG to get the E2E Alliance list information.

For the ASC, the procedure is as follows:

1. ASC connects to the REG.

2. ASC sends INFO_REQUEST message to the REG. If ASC only wants to get the update information, the Action ID Parameter in the message may be the Last Action ID got from the last INFO_REQ_ACK message received from the REG; if ASC wants to get the whole Alliance member list, the Action ID Parameter should set to be zero.

3. ASC receives INFO_REQ_ACK message or INFO_REQ_NAK message from REG.

4. ASC closes the connection.

For the REG, the procedure is as follows:

1. REG accepts the connection request from ASC.

2. REG receives INFO_REQUEST message from the ASC.

3. REG validates the AS number in the INFO_REQUEST message. If the AS number is not in the E2E Alliance member list, REG sends INFO_REQ_NAK message with error code set to be "invalid AS number", and REG closes the connection; else, go to the next step.

4. REG validates the peer address of connection. If the address does not match with the record in the E2E Alliance member list, REG sends INFO_REQ_NAK message with error code set to be "invalid ASC address", and REG closes the connection; else, go to the next step.

5. REG sends INFO_REQ_ACK message to ASC, with the requested Alliance member list information.

6. REG closes the connection.
7.3.2. REG Initiate Notification to the ASC

When some changes are made to the E2E Alliance member list at the REG, REG should send notification to the ASC of all the member in the E2E Alliance member list.

For the REG, the procedure is as follows:

1. REG connects to the ASC.
2. REG sends INFO_NOTIF message to the ASC.
3. REG closes the connection.

For the ASC, the procedure is as follows:

1. ASC accepts connection request from REG.
2. ASC validates peer address of the connection. If the address does not match with the recorded REG address (got with other means), ASC closes the connection; else, go to next step.
3. ASC receives INFO_NOTIF message from REG.
4. ASC closes the connection.
5. ASC compares the Latest Action ID in INFO_NOTIF message with the Last Action ID got from last INFO_REQ_ACK message. received. If the action ID in INFO_NOTIF message is more recent, ASC sends request to REG as described in Section 7.3.1; else, ASC makes no action.
8. Protocol Between AS Control Servers

8.1. Overview

The protocol between AS Control Servers has two major functions:

- Spread the information of local prefix list on each ASC to all other AS members.
- Synchronize the signature between each pair of AS members.

8.2. Message Formats

This section describes message formats used in the protocol between AS Control Servers. There are eight types of messages. Four types of messages are used in exchanging information on prefix list: PREFIX_REQUEST, PREFIX_REQ_ACK, PREFIX_REQ_NAK, PREFIX_NOTIF. Two types of messages are used in exchanging information on signature: SIG_OUT_NOTIF, SIG_OUT_OK. One type of messages are used in keep-alive function: SIG_KEEPALIVE.

8.2.1. PREFIX_REQUEST Message Formats

When ASC wants to get the list of prefix owned by other AS, it connects to the ASC of the AS member. After the connection is established between two ASCs, the requesting ASC sends PREFIX_REQUEST message to other ASC.

The PREFIX_REQUEST message contains the following fields:

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
|    Command    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
|     My Autonomous System      |     Peer Autonomous System    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
|                        Action ID Parameter                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```

- **Command**: This 1-octet unsigned integer indicates the type code of the message. For PREFIX_REQUEST message, the command value is 5.

- **My Autonomous System**: This 2-octet unsigned integer indicates the Autonomous System number of the sender.

- **Peer Autonomous System**: This 2-octet unsigned integer indicates the Autonomous System number of the intended receiver.
Action ID Parameter: This 4-octet unsigned integer indicates the action ID parameter for this request. It’s the latest action ID that requesting ASC getting from the peer ASC. If the ASC wants to get the information of all prefix list, the value should be set to 0.

8.2.2. PREFIX_REQ_ACK Message Formats

After one ASC receives PREFIX_REQUEST message from the other ASC, if the request is valid, the ASC will send back PREFIX_REQ_ACK message

The PREFIX_REQ_ACK message contains the following fields:

<table>
<thead>
<tr>
<th>Command</th>
<th>Number of Data Records</th>
<th>Last Action ID</th>
<th>Data Records</th>
</tr>
</thead>
</table>

Command: This 1-octet unsigned integer indicates the type code of the message. For PREFIX_REQ_ACK message, the command value is 6.

Number of Data Records: This 4-octet unsigned integer indicates the number of the total records returned.

Last Action ID: This 4-octet unsigned integer indicates the latest action ID of the records returned.

Data Records: This is a variable length field that contains a list of the prefix list information. Each prefix list info record contains the following field:

<table>
<thead>
<tr>
<th>Prefix Address</th>
<th>Prefix Length</th>
<th>Flag</th>
<th>Last Action</th>
</tr>
</thead>
</table>

Prefix Address: This 16-octet data indicates the address of the IPv6 prefix.
Prefix Length: This 1-octet unsigned integer indicates the length of the prefix.

Flag: This is 1-octet unsigned integer indicates the flag for this prefix. The meaning of the flag field for a prefix has been discussed in Section 5.3. The value of this field may be:

- 1: this prefix belongs to this AS member.
- 0: this prefix does not belong to this AS member.

Last Action: This is 1-octet unsigned integer indicates the last action for this prefix. The value of this field may be:

- Add(1): last action is "add" for this prefix.
- Delete(0): last action is "delete" for this prefix.

8.2.3. PREFIX_REQ_NAK Message Formats

After one ASC receives PREFIX_REQUEST message from the other ASC, if the request is invalid, the ASC will send back PREFIX_REQ_NAK message to the other ASC.

The PREFIX_REQ_NAK message contains the following fields:

```
+----------------+----------------+----------------+----------------+----------------+----------------+----------------+----------------+
|    Command    |  Error Code    |                  |
+----------------+----------------+----------------+----------------+----------------+----------------+----------------+
```

Command: This 1-octet unsigned integer indicates the type code of the message. For PREFIX_REQ_NAK message, the command value is 7.

Error Code: This 1-octet unsigned integer indicates the error type in processing this request. The value of the error code may be:

- 1: invalid AS number. The AS that sends the request to the ASC is not in the E2E Alliance member list at local AS.
- 2: invalid ASC address. The address of the sending host does not match the address in the E2E Alliance member list.
- 3: invalid peer AS number. The "Peer Autonomous System" field in the PREFIX_REQUEST does not match the local AS number.
8.2.4. PREFIX_NOTIF Message Formats

If there are some update to the local prefix list at the ASC, this ASC should send PREFIX_NOTIF message to ASC of all the AS member in the E2E Alliance.

The PREFIX_NOTIF message contains the following fields:

```
+--------+--------+--------+--------+
| Command|        |        |        |
+--------+--------+--------+--------+
|        | My Autonomous System |        |
+--------+----------------------+--------+
|        | Latest Action ID      |        |
+--------+----------------------+--------+
```

Command: This 1-octet unsigned integer indicates the type code of the message. For PREFIX_NOTIF message, the command value is 8.

My Autonomous System: This 2-octet unsigned integer indicates the AS number of the AS member that sending this message.

Latest Action ID: This 4-octet unsigned integer indicates the action ID of lastest update to the prefix list at the sending ASC.

8.2.5. SIG_OUT_NOTIF Message Formats

If one ASC wants to change its incoming direction signature with the other AS member, it sends SIG_OUT_NOTIF message to the ASC of the other AS member.

The PREFIX_NOTIF message contains the following fields:

```
+--------+--------+--------+--------+
| Command|        |        |        |
+--------+--------+--------+--------+
|        | My Autonomous System | Peer Autonomous System |
+--------+----------------------+------------------------+
|        | Sequence of Signature |        |
+--------+----------------------+------------------------+
|        | Signature (6 octets)  |        |
+--------+----------------------+------------------------+
```

Command: This 1-octet unsigned integer indicates the type code of the
message. For SIG_OUT_NOTIF message, the command value is 9.

My Autonomous System: This 2-octet unsigned integer indicates the Autonomous System number of the sender.

Peer Autonomous System: This 2-octet unsigned integer indicates the Autonomous System number of the intended receiver.

Sequence of Signature: This 4-octet unsigned integer indicates the sequence of the signature in this message.

Signature: This 6-octet data indicates the signature suggested by the sending ASC.

8.2.6. SIG_OUT_OK Message Formats

After one ASC receives SIG_OUT_NOTIF message from the other AS member, it should change the out-going direction signature with the other AS member. The procedure of changing of the signature (including change the configuration at edge routers) may take some time. After this procedure is finished, the ASC sends SIG_OUT_OK message to the other AS member.

The SIG_OUT_OK message contains the following fields:

0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Command    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     My Autonomous System      |     Peer Autonomous System    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Sequence of Signature                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Command: This 1-octet unsigned integer indicates the type code of the message. For SIG_OUT_OK message, the command value is 10.

My Autonomous System: This 2-octet unsigned integer indicates the Autonomous System number of the sender.

Peer Autonomous System: This 2-octet unsigned integer indicates the Autonomous System number of the intended receiver.

Sequence of Signature: This 4-octet unsigned integer indicates the sequence of the signature in this message. It is derived from last SIG_OUT_NOTIF message received.
8.2.7. SIG_KEEPALIVE Message Formats

The SIG_KEEPALIVE message contains the following fields:

\[\begin{array}{cccc}
0 & 1 & 2 & 3 \\
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 0 & 1 \\
\end{array}\]

\[\begin{array}{ccc}
\text{Command} & \text{My Autonomous System} & \text{Peer Autonomous System} \\
\end{array}\]

Command: This 1-octet unsigned integer indicates the type code of the message. For SIG_OUT_OK message, the command value is 11.

My Autonomous System: This 2-octet unsigned integer indicates the Autonomous System number of the sender.

Peer Autonomous System: This 2-octet unsigned integer indicates the Autonomous System number of the intended receiver.

8.3. Procedure for Prefix Information Exchange

The way that ASC spreads prefix information is quite similar to the way that REG spreads E2E Alliance member list information. There are two possible ways of exchanging the prefix information. First, an ASC may initiate sending a request to another ASC to get its local prefix list information. Second, when some modification is made at an ASC, it should notify all other ASC of the E2E Alliance about this modification.

8.3.1. ASC Initiate Request to Another ASC

An ASC (e.g., ASC_A) may send request to another ASC (e.g., ASC_B) to get its prefix list.

For the ASC that sends the request, the procedure is as follows:

1. ASC_A connects to ASC_B.

2. ASC_A sends PREFIX_REQUEST message to ASC_B. If ASC_A only wants to get the update information, the Action ID Parameter in the message may be the Last Action ID got from the last PREFIX_REQ_ACK message received from ASC_B; if ASC_A wants to get the whole prefix list of ASC_B, the Action ID Parameter should set to be zero.

3. ASC_A receives PREFIX_REQ_ACK message or PREFIX_REQ_NAK message from ASC_B.
4. ASC_A closes the connection.

For the ASC that receives the request, the procedure is as follows:

1. ASC_B accepts the connection request from ASC_A.

2. ASC_B receives INFO_REQUEST message from the ASC.

3. ASC_B validates the AS number in "Peer Autonomous System" field in the PERFIX_REQUEST message. If the AS number does not match the AS number of ASC_B, ASC_B sends PREFIX_REQ_NAK message with error code set to be "invalid peer AS number", and ASC_B closes the connection; else, go to the next step.

4. ASC_B validates the AS number in "My Autonomous System" field. If the AS number is not in the E2E Alliance member list, ASC_B sends PREFIX_REQ_NAK message with error code set to be "invalid AS number", and ASC_B closes the connection; else, go to the next step.

5. ASC_B validates the peer address of connection. If the address does not match with the record in the E2E Alliance member list, ASC_B sends PREFIX_REQ_NAK message with error code set to be "invalid ASC address", and ASC_B closes the connection; else, go to the next step.

6. ASC_B sends INFO_REQ_ACK message to ASC_B, with the requested prefix list information.

7. ASC_B closes the connection.

8.3.2. ASC Initiate Notification to other ASCs

When some changes are made to the prefix list at the ASC, ASC should send notification to other ASC of all the member in the E2E Alliance member list.

Suppose there is some changes at prefix list of ASC_A, and ASC_A notifies it to ASC_B.

For ASC_A, the procedure is as follows:

1. ASC_A connects to ASC_B.

2. ASC_A sends PREFIX_NOTIF message to ASC_B.

3. ASC_A close the connection.

For ASC_B, the procedure is as follows:
1. ASC_B accepts connection request from ASC_A.

2. ASC_B receives PREFIX_NOTIF message from ASC_A.

3. ASC_B validates the AS number in "My Autonomous System" field. If the AS number is not in the E2E Alliance member list, ASC_B closes the connection; else, go to the next step.

4. ASC_B validates peer address of the connection. If the address does not match with the record in the E2E Alliance member list, ASC_B closes the connection; else, go to the next step.

5. ASC_B compares the Latest Action ID in PREFIX_NOTIF message with the Last Action ID got from last PREFIX_REQ_ACK message received. If the action ID in PREFIX_NOTIF message is more recent, ASC_B sends request to ASC_A as described in Section 8.3.1; else, ASC_B makes no action.

8.4. Procedure for Signature Information Exchange

8.4.1. Overview

A pair of signatures can be set up between two AS members in the Alliance. Each signature is only used for the one direction traffic between the two ASes.

An AS member can decide whether signature is applied for its incoming traffic, and decides when to change the signature and how to change the signature. This makes one AS control its own incoming traffic. The AS that sending traffic to this AS should modify its signature for out-going traffic in response to the request from the receiving AS.

```
+----------+      Sig-21     +----------+
|          |<----------------|          |
|   AS-1   |      Sig-12     |   AS-2   |
|          |---------------->|          |
+----------+                 +----------+
```

As suggested in [Bre05], the signature used between two AS should be changed periodically. We need to design the mechanism to smoothly change the signature. Also, there may be cases that the synchronization for the signature between ASes is broken. We need to design some mechanism to detect and cure such out of synchronization.
8.4.2. Procedure of Managing the Signature

Here the procedure for managing the signature of one direction traffic is described. We’ll describe how to manage the signature for the direction from AS-2 to AS-1. ASC-1 is the AS Control Server of AS-1. AER-1 is the edge router of AS-1. Of course, AS-1 may have more than one edge routers. AER-1 is only a representative of all edge routers of AS-1. Similarly, ASC-2 is the AS Control Server of AS-1, and AER-2 is the edge router of AS-2.

```
+----------+                     +----------+
|          |        Sig-21       |          |
|  ASC-1   |<--------------------|  ASC-2   |
|          |                     |          |
+----------+                     +----------+
     /|
    / |
   /  |
  +-----+    +--|--+    +-----+
 | AER-1 |<--|--| AER-2 |
 |       |<--|--|       |
 | AS-1  |<--|--| AS-2  |
```

8.4.2.1. Start to Use the Signature

At the start, there is no signature set up for the traffic from AS-2 to AS-1. Then AS-1 decides to set up the signature. The procedure is as follows:

1. ASC-1 generates one 48-bit signature.

2. ASC-1 sends the signature configuration AER-1. The state of this signature on the edge router is Start, the edge router can accept a packet from AS-2 without any signature, or with the signature.

3. ASC-1 waits for the finish of signature configuration at AER-1. If the configuration is finished at AER-1, AER-1 sends a notification to ASC-1.

4. After the signature has been configured to all the edge routers in AS-1, ASC-1 sends SIG_OUT_NOTIF message to ASC-2.

5. ASC-2 validates the SIG_OUT message received. Then it configures the signature to all edge routers in AS-2.

6. After the signature has been configured to all the edge routers in AS-2, ASC-2 sends SIG_OUT_OK message to ASC-1.
7. ASC-1 changes the state of this signature on all edge routers to Single. The edge router can only accept packet with this signature from AS-2.

In the above procedure, the communication between two ASCs is based on TCP connection.

8.4.2.2. Periodically Change the Signature

After some time interval, ASC-1 will change the signature. The procedure is as follows:

1. ASC-1 generates a new signature.
2. ASC-1 sends the signature configuration AER-1. The state of this signature on the edge router is Switching, the edge router can accept a packet from AS-2 without the old signature, or with the new signature.
3. ASC-1 waits for the finish of signature configuration at AER-1. If the configuration is finished at AER-1, AER-1 sends a notification to ASC-1.
4. After the signature has been configured to all the edge routers in AS-1, ASC-1 sends SIG_OUT_NOTIF message to ASC-2.
5. ASC-2 validates the SIG_OUT message received. Then it configures the new signature to all edge routers in AS-2.
6. After the new signature has been configured to all the edge routers in AS-2, ASC-2 sends SIG_OUT_OK message to ASC-1.
7. ASC-1 changes the state of this signature on all edge routers to Single. The edge router can only accept packet with the new signature from AS-2.

In the above procedure, the communication between two ASCs is based on TCP connection.

8.4.2.3. Handle Out of Synchronization of the Signature

The synchronization of signature (i.e., the state of the signature at the sending AS and the receiving AS is synchronized) between two ASes may be broken. So there must be some mechanism to detect this situation and recover from this situation.

After the signature is set up at the sending AS, the ASC of the sending AS should periodically send SIG_KEEPALIVE message to the ASC.
of receiving AS. The SIG_KEEPALIVE message is sent over UDP protocol. If the receiving AS does not receive SIG_KEEPALIVE message from the sending AS for a specific time interval, the receiving AS deduces that the signature for the traffic from the sending AS to the receiving AS is broken. It should set the state of the signature to Start, and repeat the procedure specified in Section 8.4.2.1.

SIG_KEEPALIVE is only sent when the state of signature at the receiving AS is Single or Switching. In these two states, the incoming traffic is protected by the signature, so SIG_KEEPALIVE with spoofed source address can not be delivered to the ASC of the receiving AS.

8.4.3. FSM for the Incoming Direction Signature

To clarify the management of the signature, we design a Finite State Machine for the incoming direction signature.

The states and the events in this FSM are as follows:

Incoming Signature States:

1 - None
2 - Start
3 - Single
4 - Switching

Incoming Signature Sub-state (for state Start and Switching):

1 - Wait-Local-Conf
2 - Wait-Peer-Conf

Incoming Signature Events:

1 - Start command
2 - Local configuration finished
3 - Receive SIG_OUT_OK message
4 - Sig-Switching timer expires
5 - Keep-Alive detection fails

For state Start and Switching, two sub states are defined. Wait-Local-Conf is the state that ASC of receiving AS is waiting for the signature to be configured to the local edge routers. Wait-Peer-Conf is the state that ASC of receiving AS is waiting for the signature to be configured to the edge routers in the peer AS.

The transition of the state (not include the sub states) is depicted as follows:
None

---

(1)

\|--

(2, 3)  (4)

---

Start  Single  Switching

---

(5)  (2, 3)

---

The transition of the state (include sub states) is also depicted in the following table:

<table>
<thead>
<tr>
<th>Event</th>
<th>Actions</th>
<th>Message Sent</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>None (1)</td>
<td>Generate incoming direction signature.</td>
<td>none</td>
<td>2.1</td>
</tr>
<tr>
<td>1</td>
<td>Start to set signature to all edge routers of local AS.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Start(2)
Wait-Local-Conf(2.1)

<table>
<thead>
<tr>
<th>2</th>
<th>none</th>
<th>SIG_OUT</th>
<th>2.2</th>
</tr>
</thead>
</table>

Wait-Peer-Conf(2.2)

<table>
<thead>
<tr>
<th>3</th>
<th>Start sig-Switching timer</th>
<th>none</th>
<th>3</th>
</tr>
</thead>
</table>

Single (3)

<table>
<thead>
<tr>
<th>4</th>
<th>Generate incoming direction signature.</th>
<th>none</th>
<th>4.1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Start to set signature to all edge routers of local AS.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>5</th>
<th>Generate incoming direction signature.</th>
<th>none</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Start to set signature to all edge routers of local AS.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Switching (4)
Wait-Local-Conf(4.1)

<table>
<thead>
<tr>
<th>2</th>
<th>none</th>
<th>SIG_OUT</th>
<th>4.2</th>
</tr>
</thead>
</table>

Wait-Peer-Conf(4.2)
3  Start sig-Switching timer  none  3
9. Other Discussion

9.1. Coexist with IPSEC

The deployment of this method will not break the communication with IPSEC.

Though this method will change the packet at the sending AS, it will modify it back to its original packet. So it will not break IPSEC.

9.2. Difference from IPSEC

Someone may ask why we don’t build secure tunnel between two ASes with IPSEC, instead of using E2E mechanism. We agree that the basic mechanism of IPSEC and this method are similar, i.e., applying end-to-end signature to assure the identity of the source. But this method is different from IPSEC at the following aspects:

- The two ends of an IPSEC tunnel are identified by IP address, while with this method the two ends are AS networks. An AS network may have more than one entrance interface. So it is unsuitable to use IPSEC between two ASes.

- This method is used between ASes, usually with high bandwidth links, so it should be simplified. While IPSEC is still too complex for processing high bandwidth link.

9.3. The Overhead of Communication Between AS Members

The overhead of maintain E2E Alliance is not O(N^2), but O(N). Also, the traffic for exchanging the signature and the prefix for an AS member is acceptable.

Each AS may decide whether to exchange prefix and signature with other AS member. Each AS can decides whether to secure the incoming direction from one other AS member. If this AS decide to do so, it can request prefix list information from other AS member, and negotiate incoming direction signature with other AS member.

9.4. Method For Guessing the Signature

The method for guessing the signature between two AS members has been described in [Bre05]. For the convinience of the reader to understand the problem, we also describe it here.

If an attacker wants to send attacking traffic to AS-1 with spoofed source address of AS-2, he should control at least one host in AS-2. He should also control at least one host C in a network that doesn’t
deploy this mechanism. Host C can send ICMP Echo Request packet to one host A in AS-1, with the source address of host B. C can change the signature in the packet. If host B receives ICMP Echo Response from host A, the attacker can decide that the signature in the corresponding ICMP Echo Request packet is the one he wants to get.

The authors of [Bre05] suggested to use 32-bit signature. But we change to use another view-point to study this problem, and decide that 32-bit is still not enough. The attacker may control hundreds or even more hosts in the network without deployment of this method. Suppose the attacker controls 1 giga bandwidth to the destination AS (it is not very difficult for a large AS), and suppose the size of ICMP Echo Request packet sent by the attacker is 64-byte long. In the worst case, it only takes the attacker about 30 minutes to get the right signature. Considering that the link capacity will increase a lot in the near future, we suggest to use longer signature.
10. Future Work

Some issues to be worked on are:

- The design of the message is still very simple. There is possibility that transmission of message over TCP connection fails, and such mechanism like the design in BGP [RFC1654] should be added.

- Signature generation algorithm. How to generate the 48-bit signature is not decided yet.

- How to process ICMP "Packet Too Big" packets by the AS edge routers. The AS edge routers should decrease the MTU field in the ICMP "Packet Too Big" packets.
11. IANA Considerations

A new type of option in Hop-by-Hop Options Header is defined in Section 5.1.

A new type of IPv6 extension header is defined in Section 5.1.

A special TCP port should be allocated for listening at Registration Server and AS Control Server.

A special UDP port should be allocated for listen at the AS Control Server to receive Keep-alive packets.
12. Security Considerations
13. Acknowledgements

The authors would like to acknowledge the contributors who provided helpful inputs on earlier versions of this document.
14. References


Authors’ Addresses

Jianping Wu  
CERNET  
Network Center, Tsinghua University  
Beijing  100084  
China

Email: jianping@cernet.edu.cn

Jun Bi  
CERNET  
Network Center, Tsinghua University  
Beijing  100084  
China

Email: junbi@cernet.edu.cn

Xing Li  
CERNET  
Network Center, Tsinghua University  
Beijing  100084  
China

Email: xing@cernet.edu.cn

Mingjiang Ye  
CERNET  
Network Center, Tsinghua University  
Beijing  100084  
China

Email: yemingjiang@csnet1.cs.tsinghua.edu.cn
Full Copyright Statement

Copyright (C) The IETF Trust (2007).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).