Preventing IP Source Address Spoofing: A Two-Level, State Machine-Based Method

BI Jun (毕军)**, LIU Bingyang (刘冰洋), WU Jianping (吴建平), SHEN Yan (沈燕)

Tsinghua National Laboratory for Information Science and Technology, Network Research Center, Tsinghua University, China Education and Research Network (CERNET), Beijing 100084, China

Abstract: A signature-and-verification-based method, automatic peer-to-peer anti-spoofing (APPA), is proposed to prevent IP source address spoofing. In this method, signatures are tagged into the packets at the source peer, and verified and removed at the verification peer where packets with incorrect signatures are filtered. A unique state machine, which is used to generate signatures, is associated with each ordered pair of APPA peers. As the state machine automatically transits, the signature changes accordingly. KISS random number generator is used as the signature generating algorithm, which makes the state machine very small and fast and requires very low management costs. APPA has an intra-AS (autonomous system) level and an inter-AS level. In the intra-AS level, signatures are tagged into each departing packet at the host and verified at the gateway to achieve finer-grained anti-spoofing than ingress filtering. In the inter-AS level, signatures are tagged at the source AS border router and verified at the destination AS border router to achieve prefix-level anti-spoofing, and the automatic state machine enables the peers to change signatures without negotiation which makes APPA attack-resilient compared with the spoofing prevention method. The results show that the two levels are both incentive for deployment, and they make APPA an integrated anti-spoofing solution.

Key words: source address spoofing; spoofing prevention; internet security

Introduction

Routers in the modern Internet do not verify the IP source addresses of packets. Many operating systems provide application programming interfaces (APIs) to modify the IP source address of packets and many spoofing tools have been developed. These make spoofing of IP source addresses quite easy with hackers taking advantage of spoofing to threaten the Internet, such as DOS/DDOS attacks, which are the most prevalent attacks in the Internet. DOS/DDOS attacks use spoofing of IP source addresses to (1) amplify attacks such as DNS amplification attacks\(^1\)\(^2\) which will not work without spoofing\(^3\); (2) weaken the victim’s defensive ability since the victim can not distinguish the abnormal packets by their IP source address, such as in TCP SYN flooding\(^4\); and (3) conceal the real attacker, which makes law enforcement much harder. According to Moore et al.\(^5\), about 3000-4000 large-scale DOS/DDOS attacks are launched each week. Spoofing is also used in other attacks such as imputation attacks. A research report by US CERT\(^6\) showed that the rate of Internet security attacks is increasing much faster than the development of the Internet itself. These attacks which take advantage of spoofing have resulted...
in huge damages since the Internet is such an important information infrastructure.

The current IP address anti-spoofing methods could be classified into topology-based methods and peer-to-peer methods (peer-to-peer here generally refers to the source-destination model where the peer may be an autonomous system (AS) or an end system). Topology-based methods trace the real source or filter packets according to information about the Internet topology, such as ingress filtering\cite{7}, uRPF\cite{8}, and traceback. Peer-to-peer methods ignore the details of the Internet topology and only work at the source and destination, such as SPM\cite{9}, hop-count\cite{10}, authentication header (AH)\cite{11}, and Passport\cite{12}. The method proposed here is also a peer-to-peer method.

The main disadvantage of topology-based methods is that the benefit will not be significant until they are deployed on a large scale of the Internet because they need the cooperation of the whole Internet, and the AS that deploys them early can not get their benefit unless most of the other ASes deploy them. The most famous peer-to-peer method is SPM. In SPM, the router at the border of the source AS tags a key into the IP packet and the destination AS border router verifies the keys to check the packet authenticity. Thus, in this method, the ASes form an address authentication team. Members of the team can not spoof any one in the team and members not in the team cannot spoof members in the team, which bring direct benefit to the networks using SPM. However, the key-update mechanism is not attack-resilient, and is weak on a large scale. Also, SPM does not completely prevent spoofing, since such current methods cannot prevent IP address spoofing within the network, that is, attackers can spoof IP addresses in the same AS or sub-network.

This paper presents a signature-and-verification-based IP source address spoofing prevention method, automatic peer-to-peer anti-spoofing (APPA). This method takes advantage of an automatically synchronizing state machine to generate signatures without negotiation. The signature is tagged into the packet by the source and verified by the destination or router for authentication. APPA has an intra-AS level and an inter-AS level. In the intra-AS level, the end host tags a one-time key into each departing packet and the gateway at the AS border verifies the key. In the inter-AS level, the gateway at the source AS border tags a periodically changing key into each departing packet and the gateway at the destination AS border verifies and removes the keys. The most important part of the APPA is the automatically synchronizing state machine, which automatically updates the keys. The APPA system prevents all IP address spoofing so that end systems cannot even spoof addresses in the same AS or subnet with very low running and management costs, and supports anti-replay attacks and incremental deployment.

1 State Machine-Based Key Generation and Verification

1.1 Characteristics

The APPA signature-and-verification method tags a key into each packet at the source and verifies the key at the destination. The key is used in only one packet and changes in the next in the intra-AS level. This negates eavesdropping and replaying of the key or the whole packet.

Since each packet needs a unique key, a fast way is needed to produce keys rapidly at the source and verify keys at the destination. A state machine can do this very well since it has many states that transform from one to another under certain conditions. Each state is then mapped into one key by a special computation. The transition between states then changes the key. Sending of a packet causes the state transition at the source while receiving of a packet with the right key causes the transition at the destination. The APPA layout is shown in Fig. 1.

![Fig. 1 The state machine produces keys at the source and verifies keys at the destination. Each key is mapped to one state of the state machine.](image)

The state machine must satisfy several characteristics to improve the efficiency and security as follows:

(1) The state machine must have very many states...
and the periods must be very long. This makes it hard for the attackers to guess the keys or learn the whole key sequence.

(2) The same state machine should make the same key sequence. The destination can then easily use the same state machine as the source to verify the keys.

(3) The state machine must not be easily inferred from a known key sequence, which prevents an eavesdropper from verifying the state machine.

(4) It is fast and lightweight for the state machine to produce keys.

(5) The state machine must be lightweight since each source-destination pair requires a sending state machine and a verifying state machine, so the space requirement is linearly related to the number of peers.

(6) Many distinguishing state machines are required so that attackers cannot guess the state machine by brute force.

1.2 State machine design

The state machine must be fast, have a large state space and a long period, be quite lightweight, not be inferable, and have a large choice space.

One possible implementation is the Hash chain \(^{13}\) where \(key_0\) is chosen and \(key_n = f(key_{n-1})\) where \(f()\) is a Hash function such as MD-5. The source peer keeps \(key_0, key_1, ..., key_{n-1}\), and the destination peer keeps \(key_n\). The source tags \(key_{n-1}\) into the first packet and the destination verifies that \(f(key) = key_n\). If true, the destination keeps the key as \(key_n\) and the source tags \(key_{n+1}\) into the next packet. However, the period for this method is \(n\) and much space is needed; hence Hash chain is not a good implementation.

A random number generator (RNG) starts with a large number as the seed and then generates a sequence of numbers which are determined randomly. A good random number generator has a very large seed space, from \(2^{128}\) to \(2^{20000}\), and the number sequence periods are very long, from \(2^{124}\) to \(2^{20000}\). Good random number generators that are appropriate for a state machine are KISS\(^{14}\) and Mersenne Twister\(^{15}\). With a good random number generator, the seed determines the entire number sequence, so the space needed for each random number generator is only the random number generator seed. The numbers generated by the random number generator correlate to the states of the state machine. For example, the KISS generator\(^{14}\) uses a 128-bit seed to generate 32-bit numbers which are well distributed in various statistical tests, with number sequence periods of about \(2^{124}\). KISS can generate millions of numbers per second. However, two problems with KISS are that the seed is not very long and the states must generate the keys via a one-way function so that there is no practical way to infer the state from the key; otherwise, attackers may infer the seed from known keys. To solve these two problems, two random number generators can be used to generate two random number sequences. The key then comes from an XOR of the two random numbers. This is similar to the one-time-pad (OTP) cipher mechanism\(^{16}\) used in cryptography. The state machine is shown in Fig. 2.

![Fig. 2 Two random number generators are used to implement the state machine. \(a_1\) is the first number generated by generator A and \(b_1\) is the first by generator B. The pair \((a_1, b_1)\) denotes the first state of the state machine. \(key_1\) is the first key tagged into the first packet which is produced by XOR between \(a_1\) and \(b_1\). The sequence keeps a trace of how many keys are produced.](image)

This state machine must assume that the messages are all in order and there is no loss of packets. However, packets do get lost or out of order. Therefore, some additional mechanisms such as shared time period and sliding window are needed, which will be specified in the next section.

2 APPA Two-Level Solution

2.1 Inter-AS and intra-AS

In the APPA, the state machine produces a unique key for each packet but the packets may get out of order or
lost. Packets may become disordered or get lost along the route for many reasons, such as QoS or congestion. The solution is to divide the APPA into inter-AS and intra-AS levels. Some logical gateways are set inside or at the border of the AS to verify and tag the keys. In the intra-AS level, the end-system tags a unique key into each packet and the gateway verifies the authenticity of the key. The intra-AS level completely prevents spoofing in a strict way because attackers can not spoof the IP address in either another AS or the same AS. In the inter-AS level, the local gateway tags a periodically changing key into each packet and the gateway at the destination AS checks the key. The inter-AS level also checks the authenticity of the packets’ source IP prefix. The difference between the intra-AS and the inter-AS is that the key changes in each packet in the intra-AS level but only changes periodically in the inter-AS level. In an intra-AS network, eavesdropping is very easy so each packet requires a unique key to stop a hacker. However in the backbone Internet, eavesdropping rarely occurs and packet loss and disorder are more serious, which may cause the state machine to fail. Therefore, one-time-keys are necessary in the intra-AS level and also feasible with periodically changing keys in the inter-AS level if the keys change quickly. The entire APPA solution is shown in Fig. 3.

Fig. 3  The state machine produces and verifies keys both for the intra-AS (steps 1 and 2) and the inter-AS levels (steps 3 and 4). Keys are changed in each packet in the intra-AS level and changed periodically in the inter-AS level.

2.2 Inter-AS level

In the inter-AS level, the destination gateway can verify the authenticity of the packet’s IP source address prefix. Besides, it also enables the incremental deployment of APPA. The inter-AS level could be eliminated once most ASes have deployed APPA, so APPA can then prevent spoofing as efficiently as ingress filtering but much more completely. The inter-AS level has the following steps.

1. Exchange the state machine  The gateway in AS A sends its state machine (A, B) to the gateway at AS B and also receives the state machine (B, A) from B. This exchange can use some special method such as the Diffie-Hellman protocol.

2. Start synchronization  ASes A and B start the APPA at the same time. Synchronization of the state machines in the two ASes is important so it should be based on the strategy described in the following.

3. Tag and verify the key  AS A produces the key
(A to B) with the state machine (A, B) and the key (B to A) with the state machine (B, A), and then saves them in an out-table and an in-table. Then the keys used for tagging or verifying can be retrieved from the tables.

(4) **Update key** Keys are updated every 200 s since keys could accidently be revealed or found by brute-force guessing or eavesdropping. Thus, keys are changed to mitigate the effect of key disclosure.

The state machine clocks need to be carefully synchronized between every pair of ASes. A small allowed timing interval can be used to allow the state machines with the two ASes to have small errors. During this time period both the old key and the new key are accepted. If the clocks in the two ASes drift, they need to be periodically synchronized.

To recover the synchronization, each AS pair maintains two special state machines for sending and receiving. A special recovery-packet is then used for re-synchronization. The special state machine generates keys to tag the recovery-packet which is sent periodically such as every $1 \times 10^5$ s. Now, assume that ASes A and B as both APPA users and that key sequence number of key (A to B) at AS A is 16 and the timing is 120 s, while the one at AS B is 15 and 180 s. AS A sends a recovery packet with the sequence number 16 and timing 120 s to B who receives the packet and then changes the state of the state machine of AS A to re-synchronize the two ASes.

The packet format is designed according to RFC 2460[17]. The key is tagged into the packet at the hop-by-hop options header as shown in Fig. 4. The parts of the header in Fig. 4 are as follows.

**Option Type:** The three highest-order bits should be 000. The two highest-order bits indicate to “skip over this option and continue processing the header”, while the third highest-order bit indicates that the “option data does not change en-route”.

**Opt Data Len:** 6.

**APPA Type:** “00000001” for ordinary packets and “00000002” for synchronization recovery packets.

**Algorithm:** The KISS random number generator can be used with an algorithm identifier of “00000001”.

**Key:** The key is a 32-bit length string.

The edge router at the destination AS verifies the key and removes it.

![Fig. 4 Packet format for the inter-AS level key](image)

The APPA inter-AS level is similar to the SPM[9], except that SPM updates the keys by negotiation between peers while APPA updates keys automatically, which is a big difference. Updating of keys by negotiation is not attack-resilient because it costs too much and needs communication through the Internet. Each peer has to exchange keys with all the other peers in one key period, hence, the negotiation complexity is $O(n^2)$ in one key period where $n$ is the number of ASes who deploy the SPM. Thus, SPM cannot change keys very often and cannot be used in large systems. SPM also cannot be used in smaller granularity systems such as the intra-AS level. The APPA inter-AS level only needs the initial state machine exchange, with the keys then updated locally without any more negotiations. This updates keys very quickly and is also attack-resilient. Frequent key changes could significantly reduce the threat of eavesdropping. Thus, the APPA inter-AS level is much more secure than in the SPM. However, if eavesdropping frequently occurs in the backbone or there is a real-time sniffer in the backbone, periodic changes of the key would not be effective since the attackers could always know the key. However, real-time sniffing in the backbone Internet is difficult. One possible way to solve this problem is to operate a Hash computation such as MD-5 on the key together with part of the packet’s payload to produce the final key tagged into the packet. However, this would be much more expensive and probably not necessary.

### 2.3 Intra-AS level

The intra-AS level is designed to prevent spoofing within the same AS or subnet. Some attacks, for example, imputation attacks, are not yet serious threats, but may become a problem in the foreseeable future. Thus,
intra-AS level protection having the following steps is also of value.

1) **Get the IP address and the state machine**
When the end system connects to the Internet, the gateway sends a state machine which is bound to the IP address to the end system. The importance of the gateway will be discussed later.

2) **Tag key**
The end system tags a unique key into each packet. Each key is used only in one packet. Keys are generated by the state machine. Since each key is used in only one packet, it significantly mitigates the threat of sniffing but can have problems with packet loss or disorder, which would stop the state machine synchronization between the end system and the gateway. This problem is addressed in the following part.

3) **Verify the key**
The gateway verifies the key using the same state machine to make sure that the IP source address is not spoofed.

The difference between the inter-AS and intra-AS levels is that each key is used in only one packet in the intra-AS level. Thus the order of keys is very important. In the intra-AS level, where keys are tagged at the end-system and verified at the gateway, the gateway must be clearly specified. The packet loss or disorder which affects state machine synchronization is also important. The gateway is a virtual concept which may be one device or a system with several devices. The gateway is normally connected near the end systems, such as a switch or the first router near the end systems. The end system sends packets with keys and the keys are verified by this gateway. The rate of packet loss or disorder from the end system can not be serious or the system fails. A sliding window with the size of 32 can deal with the situations where the packet loss or disorder is not serious due to our experiments. This is similar to the AH\(^{11}\) of IPSec. The synchronization can be recovered using the synchronization recovery mechanism used in the inter-AS level.

The most prominent feature of the intra-AS solution is that it easily prevents replays. Since each key is used only once, a replayed key is no better than a guessed key. The 32-bit keys make replay almost impossible. This replay-prevention scheme is much better than traditional anti-replay methods such as time-stamp and sequence number.

The packet format for the intra-AS level is illustrated in Fig. 5. The parts of the header in Fig. 5 are as follows.

- **Option Type**: Same as the option type for the inter-AS level.
- **Opt Data Len**: 10.
- **APPA Type**: “10000001” for ordinary packets and “10000002” for synchronization recovery packets.
- **Algorithm**: Same as the option type for the inter-AS level.
- **State Number**: The current state number of the state machine. This is for the function of the sliding window and the current state of the synchronization.
- **Key**: The key is a 32-bit length string.

This option header is then checked and removed by the gateway.

<table>
<thead>
<tr>
<th>Option Type</th>
<th>Opt Data Len</th>
<th>APPA Type</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>State Number</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Key</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 5  Packet format for the keys in the intra-AS level

2.4 **Combination of intra-AS and inter-AS levels**

The intra-AS and inter-AS levels are combined to improve the effectiveness and to allow incremental deployment. The combined system does the following work steps.

1) The end system tags a unique key into each packet using the state machine to produce the keys.

2) The gateway checks the destination IP address of the packet to see if the destination AS has deployed APPA. If not, the packet is directly transmitted. Otherwise, the inter-AS key is produced.

3) The gateway checks the intra-AS key and discards the packet if the key is wrong.

4) The gateway tags the inter-AS key produced in step 2 into the packet and transmits it.
The destination AS gateway verifies the inter-AS packet key which changes every 200 s.

2.5 Anti-sniffing at the inter-AS level

The inter-AS level key is changed periodically, but this may not be safe enough when sniffing exists. We cannot use a unique key for each packet as in the intra-AS level because the packet disorder and loss problem is more serious in the inter-AS level. However, another method mechanism which is weaker than the one used in the intra-AS level can be used to generate new keys for each packet which is safe enough for short time periods. This solution uses a combination of the sequence number and chaos theory[18], and takes advantage of APPA's fast changing keys. The solution uses the following steps.

1. The state machine generates a key, key0, which changes periodically.
2. Each packet from source AS to destination AS contains an increasing sequence number, 1, 2, 3, 4, ... .
3. The key tagged into the packet is calculated from \( L(key_0, \text{sequence number}) \), where \( L \) is a chaos-based function, such as a four-time call to logistic mapping, \( x_{n+1} = ax_n(1-x_n), (0<x_n<1, 0<a<4) \).

In this iterative function, \( a \) is chosen by the source AS and \( x_0 = (key_0 \oplus \text{seq}) / 2^{32} \), where seq is the 32-bit sequence number. \( x_n \) is used as the key for each seq.

This mechanism is to make each key differ in each packet but the key can be easily verified by the destination AS. Thus it is not safe for long times but is okay for short times. The safety can be improved by rapidly changing key0. Since key0 is generated by the state machine, it can be changed very fast.

Logistic mapping has the butterfly effect after 3 or 4 iterations. Hence, \( L(key_0, \text{sequence number}) \) produces safe, unpredictable keys. The sequence numbers can be based on a bit map such as the bloom filter[19]. The source and destination can then easily calculate and verify the keys but the attackers can not. This mechanism is faster than MD-5.

2.6 Other considerations

The key length of 32 bits can be safe enough. The key can be tagged into the IP expansion header field in IPv6 and the IP option field in IPv4. Although APPA can be used in both IPv4 and IPv6, it is proposed for IPv6, because it is not feasible or cost-effective to change the current IPv4 infrastructure.

3 Analysis and Experiments

3.1 Feasibility and safety

In the intra-AS level, the gateway maintains the following data for each host: 1 state machine for sending, 1 state machine for receiving, and 1 state machine as a spare, where each state machine needs 256-bit seeds, a 32-bit sequence number, and 32×32 bits for the signature sliding window. The total memory is then 492 bytes for each host. An AS with 10 000 hosts then needs about 4.69 MB memory at the gateway.

Tests were conducted to verify that the new mechanism works at the intra-AS level. The tests on the Tsinghua campus network had 4 hops between the hosts and the gateway. One thousand processes on 40 computers simulated 1000 hosts sending packets using the APPA intra-AS solution. The gateway effectively handled the packets and the mechanism worked well. Large scale experiments will be done in the future.

In the intra-AS level, attackers can not replay the keys or packets because each key is used only once. Another test evaluated the safety at the intra-AS level. As shown in Fig. 6, in this test, host B replayed the key used by host A or guessed the key to spoof A's address. Host B tried \( 1 \times 10^7 \) spoofed packets with only 1 packet accepted by the gateway, so the probability of success was \( 1 \times 10^{-7} \). With the 32-bit key, the theoretical probability is about \( 9.3 \times 10^{-10} \), so the accepted packet was a very low probability event.

3.2 Implementation

The key mechanism in the inter-AS level is weaker than the one used in the intra-AS level, but is safe...
enough for short key periods. In the logistic mapping 
\( x_{n+1} = ax_n(1-x_n), \) (0\(<\textit{x}_n\)<1, 0\(<\textit{a}\)\(<4\)), \(a\) is a 32-bit or 
longer floating point number only known by the source 
AS and the destination AS. \( x_0 = (\text{key}_0 \oplus \text{seq}) / 2^{32}\), 
with \( x_0\) is then used as the key for each different se-
quence. Thus it is impossible to infer \( a\) or \( x_0\) from 
\( x_i\) in a short time. As the sequence increments, \( x_0\) 
and \( x_0\) both change and each key is used only once in 
each packet. \text{key}_0 is changed fast enough to protect it 
from attacks. The bloom filter prevents replays from 
being effective in the inter-AS level, but is not neces-
sary because it is used against real-time sniffing, which 
is believed to be very hard. Logistic mapping and state 
machine are significant to cope with general eaves-
dropping in the inter-AS level.

APPA is much safer than SPM since the SPM key 
update cannot be very fast because of the negotiations. 
Thus, SPM is not stable and may become the target of 
a DOS attack. APPA updates the keys in a fast, stable 
manner so that each packet has a unique key to prevent 
eavesdropping. In addition, SPM cannot stop spoofing 
in the same AS.

3.2 Performance

The prototype of APPA was implemented based on 
the click modular router\cite{20}. Tests were run on the to-
poIogy illustrated in Fig. 7. All the routers, gateways, 
and hosts were running Linux, PM1.6 CPU, 1 GB 
memory PCs. The routers and the gateways could 
choose to deploy different anti-spoofing strategies. The 
hosts could be configured as either legitimate users or 
attackers.

Three tests were conducted using three anti-spoofing 
strategies, SPM, inter-AS APPA, and intra-AS APPA. 
The two-level APPA was not included because the 
inter-AS level could be canceled if the intra-AS level 
was fully deployed. In the tests, the hosts sent packets 
to other hosts in either the local AS or the external 
ASes. Most hosts sent legitimate packets with their 
own addresses, with only one host employed as an 
attacker to regularly send spoofing packets. Both the 
effectiveness and performance are presented in Table 1. 
The first column lists the spoofing behaviors. For ex-
ample, “A1(B1)→C1” means that A1 pretends to be B1 
and attacks C1, i.e., A1 sends packets to C1 using B1’
’s source address. The data in the other cell are the effec-
tiveness and the performance. For example, in the first 
case with SPM up to 99.9999999% of the spoofed

![](image)
packets are filtered by CR (the edge router of AS C), and the peak value of legitimate flow throughput at CR is 201.4 MB/s. The gateways or routers can handle a maximum of 400 MB/s with no anti-spoofing strategies.

<table>
<thead>
<tr>
<th>Spoofing attack</th>
<th>CR with SPM</th>
<th>CR with inter-AS APPA</th>
<th>AG with intra-AS APPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1(B1)→C1</td>
<td>&gt;99.999 999 9</td>
<td>201.4</td>
<td>212.9</td>
</tr>
<tr>
<td>A1(A2)→C1</td>
<td>&lt;0.000 000 01</td>
<td>204.1</td>
<td>213.7</td>
</tr>
<tr>
<td>A1(arbitrary)→A2</td>
<td>0**</td>
<td>241.3</td>
<td>247.8</td>
</tr>
<tr>
<td>A1(arbitrary)→A2</td>
<td>0**</td>
<td>241.3</td>
<td>247.8</td>
</tr>
</tbody>
</table>

Note: The spoofing attack X(Y)→Z means that host X uses Y’s address to attack Z.

The intra-AS APPA can deal with all of the spoofing because it uses fine-grained real source address verification. The legitimate flow throughput is much higher because the gateway forwards the packets with both source and destination addresses in the local AS, while the edge router does not. In the intra-AS level, the state machine at the end system can produce 2 million keys per second, hence, it does not affect the end system’s bandwidth. At the local gateway, APPA gets the destination IP address from the packet, looks up the IP prefix in a table, gets the IP source address, and then gets the source address from another table and verifies it. Finally, APPA removes the key.

The effectiveness and performance of SPM and inter-AS APPA are similar, and the only difference is the key update mechanism (SPM uses negotiation while APPA uses the state machine). The effectiveness of the inter-AS APPA is almost the same as SPM and the performance is better. More importantly, the security is much better, because the key updating is much faster than in SPM. APPA is a little more complex than SPM, but the added cost is not significant. The state machine does the key updating in the local machine, without negotiating with other peers across the Internet, and the KISS random number generator makes the key generation very fast.

The performance is somewhat limited since all the routers and gateways were implemented on Linux PCs. Hardware implementation would be much faster.

4 Conclusions

This paper presents a signature-and-verification-based IP source address anti-spoofing method, APPA. A state machine is used to automatically produce and verify keys. APPA has two levels. In the intra-AS level, APPA produces a unique key for each packet which could prevent spoofing in the same AS and prevent replay attacks. In the inter-AS level, APPA changes keys frequently and makes itself attack-resilient. APPA is lightweight and supports incremental deployment, hence it can be used to completely prevent IP source address spoofing.

It is suggested that the future work should focus on designing the whole protocol and improving the APPA performance to make it safer and faster.

References


