J. Parallel Distrib. Comput. 72 (2012) 1215–1225

Contents lists available at SciVerse ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

An incrementally deployable path address scheme

MyungKeun Yoon[a], Shigang Chen[b,*]

[a] Department of Computer Engineering, Kookmin University, Seoul 136-702, Republic of Korea
[b] Department of Computer Science, University of Florida, Gainesville, FL 32611, USA

ARTICLE INFO

Article history:
Received 13 June 2011
Received in revised form
20 September 2011
Accepted 2 May 2012
Available online 15 May 2012

Keywords:
Internet protocols
Path address
Network security

ABSTRACT

The research community has proposed numerous network security solutions, each dealing with a specific problem such as address spoofing, denial-of-service attacks, denial-of-quality attacks, reflection attacks, viruses, or worms. However, due to the lack of fundamental support from the Internet, individual solutions often share little common ground in their design, which causes a practical problem: deploying all these vastly different solutions will add exceedingly high complexity to the Internet routers. In this paper, we propose a simple generic extension to the Internet, providing a new type of information, called *path addresses*, that simplify the design of security systems for packet filtering, fair resource allocation, packet classification, IP traceback, filter push-back, etc. IP addresses are owned by end hosts; path addresses are owned by the network core, which is beyond the reach of the hosts. We describe how to enhance the Internet protocols for path addresses that meet the uniqueness requirement, completeness requirement, safety requirement, and incrementally deployable requirement. We evaluate the performance of our scheme both analytically and by simulations, which show that, at small overhead, the false positive ratio and the false negative ratio can both be made negligibly small.

1. Introduction

After 30 years of accumulative development, the Internet is full of security challenges: address spoofing, denial-of-service (DoS) attacks, denial-of-quality (DoQ) attacks, reflection attacks, viruses, worms, to name a few. The research community has proposed numerous detection/mitigation solutions [10,19,17,11,4,25,26,5,9], each dealing with a specific problem in its own unique way. The vast solution space, if viewed as a whole, seems able to handle many security problems, but deploying all these solutions can be practically infeasible. A major obstacle is that individual solutions often share little common ground in their design. Many solutions heuristically work around the limitations imposed by the legacy Internet protocols and demand orthogonal changes on routers. Their combined complexity added to the Internet routers will be exceedingly high. In this paper, we take a different angle to study Internet security. We ask the following question: Can we identify a simple extension to the Internet protocols, which will provide certain new information that will assist us to solve many security problems? Such information, once created inside the network, will be made accessible at the network edge, allowing various security applications to be developed there. With most complexity remaining at the edge, *the network core, which provides application-independent information, can be kept simple and stable*.

What is the new information that the network can provide to assist the development of security applications at the Internet edge? There can be many. The one we propose here is called the *path address*. A host on the Internet is identified by an IP address; a routing path on the Internet will be identified by a path address. The big question is, can path addresses help us in ways that IP addresses cannot? Below, we use a few examples to illustrate their differences.

In the first example, suppose that a server under DoS attack attempts to identify the IP addresses of flooding sources and block the packets carrying those addresses. However, this approach will fail if malicious packets carry forged source addresses or a reflection attack is used to cover the true sources. In the second example, imagine that a server under DoQ attack tries to distribute its processing capacity fairly among the clients. It cannot perform such distribution based on IP addresses because there are too many of them. A certain kind of aggregation will be necessary. In the third example, suppose that a victim has managed to capture an attack packet (say, containing a virus). Based on this single packet, before triggering law enforcement actions, how can the victim trace *across the Internet* back to the attacker, given that the source address in the packet may be a forged one? All the above problems cannot be reliably solved based on IP addresses in the packet header, which are set by the sender and may not be genuine. We need address information that is beyond the reach of end hosts. This new address

* Corresponding author.
E-mail addresses: mkyoon@kookmin.ac.kr (M. Yoon), sgchen@cise.ufl.edu (S. Chen).

should be set and verified by the routers in the network. If each routing path is assigned a path address, which is carried in the header of packets routed on the path, then a server under DoS attack can block packets based on path addresses that identify attack paths, a server under DoQ attack can distribute its capacity among packet groups classified based on path addresses, each representing an aggregate of client traffic, and a victim can use the path address to find out the attack path and therefore the attack source at the end of the path.

We propose an incrementally deployable path address scheme (PAS) that meets the following requirements. (1) Each routing path to a certain destination has a unique path address (with very high probability), which is called the *uniqueness requirement*. It ensures that path addresses accurately point out where packets are coming from. Blocking a path address filters out the packets from an attack source without causing significant collateral damage. (2) Each packet carries the address of the path it traverses; the packet has to carry that address from the first router all the way to the destination, which is called the *completeness requirement*. It gives the flexibility of classifying or blocking packets of a given path address anywhere along the path. (3) The path address in a packet's header can only be correctly set by the routers in the network; a host will not be able to forge the path address carried in its packets without being caught, which is called the *safety requirement*. (4) Any viable path address scheme must support incremental deployment on the existing Internet. It should bring benefit when only some of the routers are upgraded for path addresses, which is called the *incrementally deployable requirement*.

This paper describes in detail how Internet protocols can be enhanced to include path addresses based on the above requirements. We demonstrate that the proposed PAS satisfies the self-completeness property for incremental deployment, so that domains will enjoy full protection as soon as they deploy the PAS. We address the problem of verifying the authenticity of path addresses. The PAS incurs small maintenance overhead. We evaluate the performance of the PAS both analytically and by simulations, which show that the false positive ratio and the false negative ratio can be both made negligibly small.

The addition of path addresses requires relatively small changes in Internet protocols. On the other hand, it may potentially have a large impact on how security systems will be designed. When a victim's intrusion detection system identifies malicious packets, it may extract the path addresses from the packets and pay special attention to future packets carrying the same path addresses, or even block such packets. If the victim has a mapping table between path addresses and source IP address prefixes, which can learn from the normal packets received in the past, then it can trace back to the source domain based on the path address of a captured attack packet. Path addresses can also make the pushback mechanism [13] more powerful. After the victim identifies a set of path addresses from malicious packets, it may push these addresses into the network for blocking. When the first-hop router receives a packet from a neighbor router and finds that the packet carries a blocked path address, it drops the packet and then pushes that path address to the neighbor router. Eventually the addresses will be pushed all the way back to the edge of the domains where the attack hosts reside.

While the focus of this paper is on network security, the path address scheme can be used in other network functions, such as packet classification, resource reservation, and service differentiation. For example, instead of per-flow queuing, packet queues can be queued based on path addresses, allowing trunk resource reservation to be made for all flows sharing the same path between two domains.

The rest of the paper is organized as follows. Section 2 discusses the related work. Section 3 presents the design of our path address scheme. Sections 4.1 and 4.2 evaluate the PAS by analysis and simulations, respectively. Section 5 draws the conclusion.

2. Related work

The research community has proposed numerous solutions to prevent address spoofing, mitigate DoS attacks, filter malicious packets, control access to critical resources, and perform trackback. While tremendous progress has been achieved, most solutions have their own limitations. More importantly, these solutions are designed based on vastly different mechanisms and, as an aggregate, they will impose enormous complexity on the Internet. Hence, to make them practically viable, we must seek ways to reduce such complexity with new generic assistance from the network. We believe that the path address is a good candidate for achieving this goal. Below we survey the related work, many of which may benefit if path-address information becomes available.

Research on preventing address spoofing has brought a number of technical breakthroughs [14]. Ingress filtering [8] requires the edge routers of stub networks to inspect outbound packets and discard those packets whose source addresses do not belong to the local networks. Realizing ingress filtering in multihomed IPv6 networks is studied in [5]. Cryptographic cookies [3] allow a server to stay stateless until the address of a client is verified. One problem is that it is more expensive for the server to generate/verify cookies than the attacker to forge request packets. The client-puzzle solutions [10,22] require clients to solve cryptographic puzzles before their connections are established. However, significant computation overhead is placed not only on malicious hosts but also on legitimate clients. The route-based packet filtering scheme [17] requires each router to drop packets that are not supposed to pass a link. The spoofing prevention method (SPM) [4] requires pairwise secure communication channels among ASes to synchronize their keys.

Many systems have been proposed to mitigate DoS attacks. Adaptive early packet filtering [6] is proposed for defending firewalls against DoS attacks. QoSoDoS [9] is a protocol that ensures delivery of time-sensitive messages over unreliable networks, susceptible to high congestion and network flooding DoS attacks. SOS [11] is a secure overlay service designed to protect emergency services from DoS attacks. Mayday [1] is a generalization of SOS. They both assume a closed group of trusted clients. WebSOS [15] applies the SOS architecture to the web service using graphic Turing tests.

IP traceback has been intensively studied [19,2,20,25,16]. The goal is to find the origins of the packets with spoofed source addresses. Many traceback schemes incur considerable computation overhead [19], storage overhead [20], or communication overhead [2] in order to keep track of the routers that the packets traverse. Moreover, they identify the attack paths but do not provide a means for the victim to block the attack packets. Recently, entropy variations between normal and attack traffic have been used to determine which network flows belong to DDoS attacks [28,24]. Each router measures the entropy variation per flow, and decides the neighboring upstream routers from which attack packets come.

The most related work is Pi (path identifier) [27,26], which requires each router to insert an n-bit mark in the IP identification field, where n is typically 2. The mark inserted by Pi in the packet header is not suitable to serve as a path address. In particular, Pi does not satisfy the uniqueness, completeness, and safety requirements (defined in Section 1). First, in Fig. 1(a), for packets coming from a long path, marks inserted by remote routers will be overwritten due to the limit size of the path identifier. Consequently, all packets arriving at R_8, even though they may come from different paths further upstream, will have the same path identifier when they reach the receiver. This violates the uniqueness requirement. To block an attacker behind R_9 based on the path identifier, all normal users behind R_8 will also have to be blocked. Second, marks are inserted one at a time by the

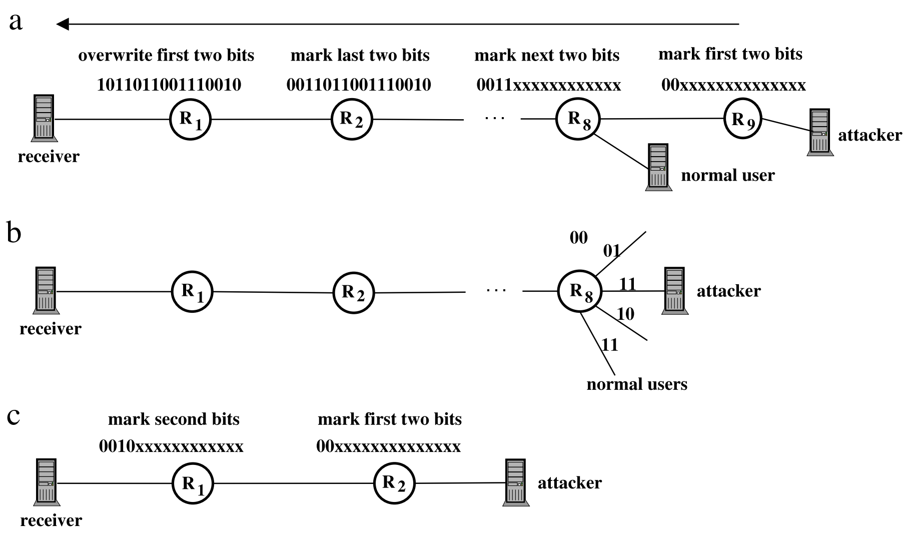

Fig. 1. Pi cannot be used for path addresses.

intermediate routers. Hence, a packet will not carry the same address information in the IP identification field along its route, which violates the completeness requirement. Third, in Fig. 1(b), if a router (such as R_8) has more than 2^n links, then there will not be enough mark values to uniquely distinguish where packets are from. On the other hand, if a router has fewer than 2^n links, it will leave some mark values unused. Fourth, in Fig. 1(c), if a zombie host is close to the receiver, only a few bits in the path identifier will be marked by routers, and the remaining bits will carry arbitrary values set by the attacker, which violates the safety requirement. To block the attacker, the receiver has to block all path identifiers that carry the same value in those few bits, which means that one sixteenth of all normal traffic will be mistakenly blocked in this example. If the zombie is one hop away from the receiver, then one fourth of all normal traffic will have to be mistakenly blocked. The problem is very serious because a *single* zombie close to the receiver can cause such significant collateral damage.

3. Path address scheme

This section presents the detailed design of the path address scheme.

3.1. Objectives

We are only concerned with the interdomain routing paths at the AS level.[1] Because the discussions are exclusively about interdomain subjects, we will sometimes refer to an "interdomain router" (e.g., a BGP router) simply as a "router" and an "interdomain routing protocol" (e.g., BGP) as a "routing protocol". We will use "AS" and "domain" interchangeably.

We propose a path address scheme (PAS), which assigns each path an address. There is an inherent difference between IP addresses and path addresses. The IP addresses are owned by the hosts, which are given the full responsibility of setting the source addresses in their packets. The path addresses are owned by interdomain routers and kept secret to the hosts. Therefore, only routers are able to set path addresses appropriately in the packet header.

We will answer the following questions: How do we define the address of a routing path? How do we extend the routing protocols to keep track of the path addresses? What new fields should be introduced in the packet header for path addresses? How can the receiver verify the authenticity of the path address carried in a packet?

A packet carrying the authentic address of its routing path is called a *normal packet*; a packet carrying a false path address is called an *abnormal packet*. Our goal is to enable the receiving host, as well as the intermediate routers, to classify the packets into these two categories. To fulfill this goal, the design of path addresses should meet the following objectives.

- *Objective* 1: *All legitimate packets will carry the authentic path addresses and therefore be classified as normal packets.*
- *Objective* 2: *All malicious packets will either carry the authentic path addresses or otherwise be classified as abnormal packets.*

The second objective needs more explanation. An attack host may inject malicious packets into a routing path. It has two choices, falsifying or not falsifying path addresses in the packet header. If the attack host sets false path addresses in its packets, the false addresses will be detected and the packets will be classified as abnormal ones. If the attack host lets the router set the authentic path address, all its packets will share a common characteristic, the same path address, which can be used for traceback or packet filtering.

3.2. Definition of path address

Each interdomain router generates a random number, called the *local number*, which has l bits. The path address of a routing path is defined as the XOR of the local numbers of the routers on the path. An example is given in Figs. 2 and 3, where only the first eight bits of the local numbers are shown. Fig. 2 shows the AS-level topology and the local numbers of nine interdomain routers. We denote the local number of a router Rx, $x \in [1 \cdots 9]$, as $Rx.loc$. Fig. 3 shows the addresses of the routing paths to $AS1$. We denote the address of the routing path from Rx to a domain y as $Rx.paddr(y)$, which is called the *path address from Rx to y*.

In Fig. 3, the path from $R1$ to $AS1$ contains only one router, $R1$. Hence, $R1.paddr(AS1) = R1.loc = 10101101$. The path from $R2$ to $AS1$ contains two routers, $R2$ and $R1$. Hence, $R2.paddr(AS1) = R1.loc \oplus R2.loc = 10101101 \oplus 00010111 = 10111010$. The path addresses of all other routing paths to $AS1$ are similarly determined.

[1] Technically, a similar scheme of path addresses may be introduced at the intradomain level, especially for large ASes.

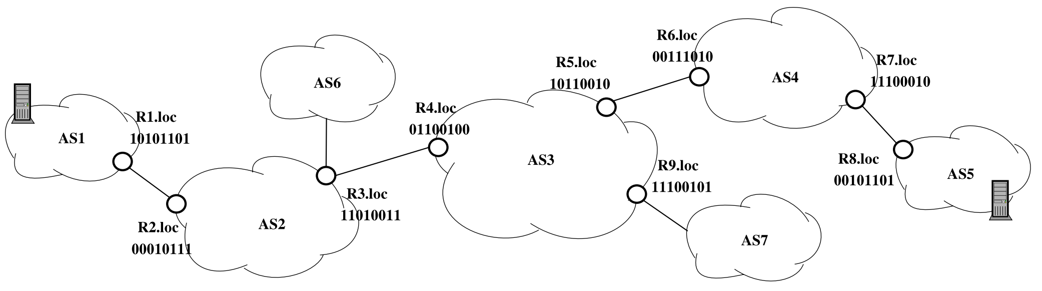

Fig. 2. The local numbers of the interdomain routers.

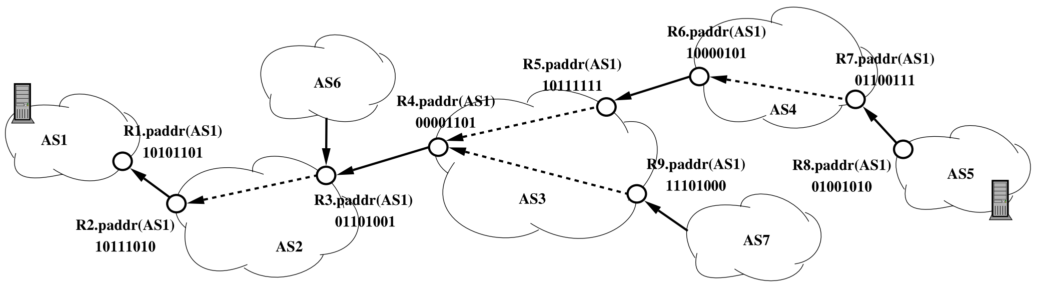

Fig. 3. The addresses for the routing paths from the routers to $AS1$. For example, $R8.paddr(AS1) = 01001010$. It is the XOR of all local numbers on the routing path $R8 \to R7 \to R6 \to R5 \to R4 \to R3 \to R2 \to R1$. Alternatively it can be viewed as the XOR of $R8$'s local number and $R7.paddr(AS1)$.

The local numbers are independent random numbers. Any two (undirected) paths have at least one different router.[2] Their addresses have at least one different local number in the XOR calculation. Therefore, the path addresses are also independent random numbers. Given a destination domain, we want the path addresses from all routers to be different with high probability, as illustrated in Fig. 3, where $R1.paddr(AS1)$ through $R9.paddr(AS1)$ are all different.

Because the path address will be carried in the packet header, its length represents a performance/overhead tradeoff. Let p be the number of bits in a path address. Suppose that a victim server decides to temporarily block a path address carried by identified malicious packets in a SYN-flood attack. Let x the number of ASes on the Internet. The expected number of other ASes whose routing paths to the victim happen to have the same address is bounded by $\frac{x}{2^p}$, and the expected fraction of legitimate packets that are mistakenly blocked is 2^{-p}, which is only related to the value of p and not related to the number of ASes. On today's Internet,[3] $x \leq 2^{16}$, and the above two numbers are 2^{-16} and 2^{-32}, respectively, if $p = 32$. We expect the value of p to be set reasonably large.

3.3. Extending the routing protocol for path address

The interdomain routing protocol (BGP) establishes a routing table at each router Rx, which has an entry for every reachable domain y. A path address field is added to the routing entry, storing $Rx.paddr(y)$, the address of the current routing path to y. The routing protocol can be easily extended to keep track of the path addresses as the routes change. First, the routers nearest to a destination y know the path addresses, which are simply their local numbers. Second, consider an arbitrary router Rx, and let Rz be the next hop on its routing path to y. If Rz knows the correct value of $Rz.paddr(y)$, Rx will be able to calculate its path address to y by $Rx.paddr(y) = Rz.paddr(y) \oplus Rx.loc$. Therefore, by induction, the correct addresses for all paths will be found after the routes stabilize, assuming that the neighboring routers exchange their knowledge about path addresses together with the classical BGP routing information. We also want to point out that the above distributed computation of path addresses does *not* assume symmetric routing.

The additional overhead (one integer for each routing entry) is very small, compared with what today's BGP already stores: the whole interdomain path to a destination domain for each routing entry.

Incremental deployment can be achieved as follows. Define a new *optional transitive path attribute* in BGP for path address. For a BGP router that is upgraded to support the PAS, when it advertises its routes to neighbors via UPDATE messages, it inserts the new transitive attribute in UPDATE to carry the path address of each route. When a BGP router that does not support the PAS receives such UPDATE messages, according to the protocol of BGP [18], it will pass the received transitive attribute (i.e., path addresses) to its neighbors when the received routes are advertised. When a BGP router that supports the PAS receives UPDATE messages with the new transitive attribute, it will extract the path addresses, and update the routing table for new routes and new path addresses (by XORing the received path addresses with the local number). When it advertises the new routes, it inserts their path addresses as transitive attribute. Under incremental deployment, the address of a path calculated by the above protocol will be the XOR of the local numbers of the routers that have been upgraded to support th PAS. The legacy routers are left out of the distributed calculation process.

A multi-homing domain has more than one path to each destination. Each of its BGP routers stores a different set of routes.

[2] There may be more than one communication link between two neighboring routers. If two paths have the same sequence of routers but use different links, we treat them as the same path.

[3] The AS identifier in BGP is 16 bits long, though it has been recently proposed to extend that to 32 bits (RFC 4893). Hence, the number of ASes on today's Internet is bounded by 2^{16}.

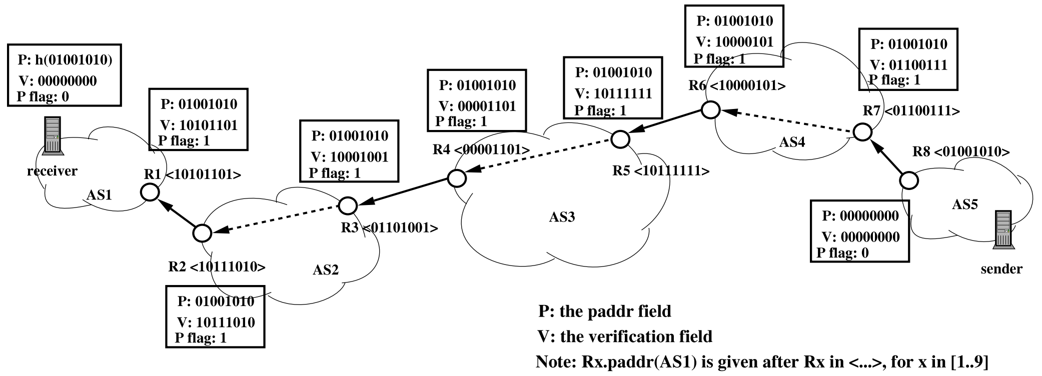

Fig. 4. The received values of the paddr and verification fields are shown beside each router. The two fields are set to 0 by the sender. The first interdomain router sets these fields with appropriate values. The path address field stays unchanged at the subsequent hops, but the verification field is XORed by the local number at each hop. The verification field should be 0 when the packet reaches its receiver.

The traffic from this domain may be split among those BGP routers, carrying different path addresses and following different routes to a destination. If an attack (such as DoS) is launched from this domain, the victim will identify more than one (most frequently appearing) path address associated with the attack.

BGP stability is of primary importance to the overall stability of the Internet [12]. Route flap damping techniques have been proposed and implemented to stabilize interdomain routes [21]. Although route change is infrequent among BGP routers overall, it does happen occasionally due to link failure or other reasons. Before new routes are stabilized, some path addresses may be temporarily out of sync, causing a burst of packets to be classified as abnormal. Therefore, not all abnormal packets should be dropped automatically. Only when a victim is under attack and the need to immediately block out malicious packets outweighs the collateral damage due to the *small possibility* of ongoing interdomain route change may the victim decide to block out all abnormal packets. Even when such misblocking happens, it is temporary. We want to point out that route change also poses similar challenge to Pi [27], IP trackback [19,2,20,27], and other related work [17].

3.4. New fields in packet header and path address verification

The PAS places two new fields and one new flag in the packet header: the *paddr field*, the *verification field*, and the *P flag*. The paddr field carries the address of the routing path that the packet traverses. For example, in Fig. 3, the packets from $AS5$ to $AS1$ will carry 01001010 in the paddr field if they are routed via $R8$. To routers, the P flag indicates whether the paddr field has been appropriately set or not. The purpose of the verification field is to prevent a malicious host from falsifying the path addresses. In Fig. 3, a malicious host in $AS4$ may forge packets with 01001010 in the paddr field and pretend that the packets are coming from $AS5$. When the forged packets arrive at $R6$, they are mixed with the legitimate packets from $AS5$, so $R6$ must be able to classify the forged ones as abnormal packets. This is accomplished with the help of the verification field. We will explain the actual operations shortly. The problems of where to place these fields and how to operate under incremental deployment will be addressed at the end of this subsection.

The source host does not know the path address of its routing path. It sets the P flag to 0, and sends the packet, which arrives at the first interdomain router. When the router receives a packet, if the P flag is 0, the router knows that it is the first hop on the path and is responsible for assigning the appropriate values for the paddr/verification fields. After that is done, the router will change the P flag to 1, so that the subsequent routers will not change the path address carried in the packet, which satisfies the completeness property.

An example is given in Fig. 4, where the received packet is shown beside the router. When the first interdomain router, $R8$, receives the packet, it finds that the P flag is 0. $R8$ sets the paddr field to be $R8.paddr(AS1)$, which is the path address from itself to the destination. It sets the verification field to be $R8.paddr(AS1) \oplus R8.loc$, which gives the path address from the next hop router to the destination. Finally it sets the P flag to 1 before forwarding the packet. When the next-hop router, $R7$, receives the packet, it keeps the path address field intact but updates the verification field by XORing it with the local number. The new value of the verification field is the path address from the yet next hop ($R6$) to the destination. Consequently, each intermediate router Rx is able to verify the authenticity of the path address in the paddr field by matching the received value in the verification field against $Rx.paddr(AS1)$, which can be found in the routing table. If the two match, then the packet is a normal one. Otherwise, it is classified as an abnormal one. The verification process must be carried out at each hop, because forged packets may be injected anywhere.

When a packet reaches the receiver, the verification field is zero if it is a normal packet and non-zero if it is an abnormal packet. Because the path address should be kept secret from the end host, the last hop router will disguise the path address by performing a keyed hash on the paddr field. All normal packets traversed the same routing path will have the same hash value in the paddr field when reaching the receiver.

So far we have described normal behavior. Next, we study what a malicious host can do. In Fig. 5, a malicious host resides in $AS4$. When producing attack packets, it has two choices, either setting the P flag to 0 or to 1. (1) If it sets the P flag to 0, $R6$ will insert the path address in the packet header. Because the packets carry the correct path address, they will be classified as normal (by definition) all the way to the receiver. When the receiver finds itself is under attack, it tries to mitigate the attack by filtering out the malicious packets. Recall that the last hop router will hash the paddr field. Without knowing the actual path address, the receiver performs filtering based on the hashed path address carried in those packets. (2) To hide itself, a malicious host may set the P flag to 1 and the paddr field to an arbitrary value. However, it does not know the correct value for the verification field, which must be the path address from $R6$ to the destination. If it sets this value incorrectly, all intermediate routers will classify the packets as abnormal. An example is given in Fig. 5, where both paddr and verification fields of an attack packet are initially set to 01001010, representing a false source of $AS5$. The packet is classified as abnormal by all routers on the path.

With extension headers, IPv6 is designed to allow easy incorporation of future functions. The new path-address fields can

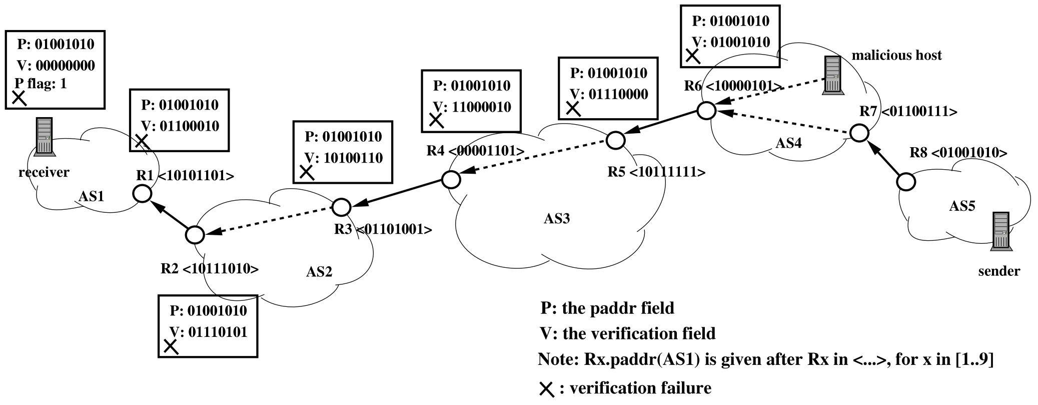

Fig. 5. A malicious host in $AS4$ sets the paddr/verification fields arbitrarily with the P flag being 1. As long as it does not know $R6.paddr(AS1)$, the attack packets to $AS1$ will be classified as abnormal, which is indicated by a cross below V in the figure.

be easily incorporated into IPv6 by adding an extension header. The new fields may also be embedded in the IPv4 header for backward compatibility by creating a new IP option or using the 16 bits from the IP identification field, 1 bit from the flag field, and 13 bits from the offset field, as many other works [27,19,2,20] do. The verification field can be made shorter than the paddr field, which provides flexibility of making a performance tradeoff under space constraints. In this case, only a subset of paddr bits are verified.

During incremental deployment, some routers are upgraded to support the PAS, while others are not. The former will process the packets as described above. The latter will simply forward the packets without PAS-related operations. Because the path address is the XOR of the local number of the upgraded routers, the verification process will be performed successfully.

Proving that the PAS satisfies the requirements of uniqueness, completeness, safety, and incremental deployment is quite straightforward, because its design is specifically constructed based on these requirements. We omit the proof to save space.

3.5. Alternative version of path address against router compromise

A serious problem arises when the attacker compromises core interdomain routers that route and forward traffic for many other ASes. It is unlikely that this problem will happen frequently, because core interdomain routers, as critical infrastructure, are closely watched. But if an attacker gains the control of a core interdomain router, it can do a variety of types of harm, such as causing inconsistent routing tables, producing false routes, and injecting forged packets. Our focus is on path address. In Fig. 5, if the malicious host compromises $R6$, it knows the path addresses from $R6$ to all destinations. The malicious host can instruct $R6$ to forge packets with arbitrary values in the paddr field but correct values in the verification field, which allows the packets to pass the verification along the routing paths. Router compromise poses similar challenge to Pi [27], IP trackback [19,2,20], and other related work [17], even though most did not consider this issue. There is no way one can save the legitimate packets arriving at the compromised $R6$ because $R6$ can corrupt or even drop them. But it is possible for us to enhance the design of path addresses so that packets from $R6$ can be separated from packets forwarded on other paths. The basic intuition is that, if the subsequent routers after $R6$ are not compromised, they should construct a portion of the path address that is beyond the control of $R6$. If necessary, this portion of the address can be used by the receiver to classify the packets from $R6$.

The new way of constructing a path address performs *shifted XOR*, instead of XOR, on the local numbers of the routers. Let the routing path from Rx to a destination domain y be $Rx \to R[x-1] \cdots \to R1 \to y$. The distance from Rx to y is x. Let d be a small integer. To calculate $Rx.paddr(y)$, when we XOR the local numbers, we shift $R2.loc$ to the right by d bits, $R3.loc$ to the right by $2d$ bits, . . ., and $Rx.loc$ to the right by $(x-1)d$ bits. Hence, the enhanced version of path address is defined as follows.

$$\begin{aligned} Rx.paddr(y) &= \bigoplus_{i\in[1\cdots x]} (Ri.loc \gg (i-1)d) \\ &= R[x-1].paddr(y) \oplus (Rx.loc \gg (x-1)d), \qquad (1) \end{aligned}$$

where $\gg$ is the right shift operator. It is easy for a routing protocol to keep track of the new path address if Rx knows its distance to a destination y and knows the path address from its next hop (denoted as $R[x-1]$) to y. Below, we give a few examples based on the local numbers in Fig. 2. Let $d = 2$.

$$\begin{aligned} R1.paddr(y) &= R1.loc = \mathit{10}101101\cdots \\ R2.paddr(y) &= R1.paddr(y) \oplus (R2.loc \gg 2) \\ &= 10101101\cdots \oplus \mathbf{00}000010111\cdots \\ &= \mathit{1010}1000\cdots \\ R3.paddr(y) &= R2.paddr(y) \oplus (R3.loc \gg 4) \\ &= 10101000\cdots \oplus \mathbf{0000}11010011\cdots \\ &= \mathit{101001}01\cdots, \end{aligned}$$

where the bold zeros are inserted due to right-shift operations, and the italic bits in a path address will not change in the subsequent computations. The leftmost d bits of a path address are determined by the last hop ($R1$) on the routing path, the next d bits are determined by the last two hops, and so on. If Ri is compromised, it has no impact on the leftmost $(i-1)d$ bits in the path address.

The procedure for setting the values in the paddr/verification fields is similar to what has been described in Section 3.4, except that shifted XOR is used. The verification is, however, different. When an intermediate router Ri receives the packet, it classifies the packet as normal only if the verification field matches $Ri.paddr(y)$ and the leftmost $(i \times d)$ bits of the paddr field match those in $Ri.paddr(y)$.

If a malicious host compromises Ri, it cannot set the leftmost $(i-1)d$ bits in the paddr field to arbitrary values, because they have to match those in $R[i-1].paddr(y)$ in order to pass the verification of the next hop. Consequently, all attack packets from Ri will carry the same $(i-1)d$ bits in the paddr field. A defense system may be designed based on this property. What happens if the attacker compromises $R1$, the last hop router to the destination? Because the packets from all over the Internet are fully mixed there, it is no longer possible to separate the legitimate packets from the malicious ones if that router is compromised, unless the legitimate packets are protected by end-to-end cryptographic schemes.

Shifted XOR shares superficial similarity with the operation of Pi [27]. In Pi, each router can only set two bits in the IP identification field. A malicious host that is one hop away from the victim can arbitrarily set other bits in that field. In shifted XOR, each router has much more impact. For example, the last hop router will influence all bits in the path address. The malicious host one hop away cannot set any bit in the path address without being detected.

3.6. Self-completeness of the PAS for incremental deployment

During incremental deployment, let C be the set of ASes that have deployed a defense system and C' the set of ASes that have not. The system is said to be *self-complete* if it is fully functional among ASes in C even when attacks are launched from C'. A self-complete system must defeat both the "internal" attackers from C, which are within the defense coverage, and the "external" attackers from C', which are outside the defense coverage.

Many existing defense systems are not self-complete. Take ingress filtering [8] as an example. Suppose that all networks in C perform ingress filtering and that those in C' do not. The attackers from C' can forge any source addresses and pretend to be from C. A victim cannot distinguish such attack packets from legitimate packets from C, and has to drop both. Hence, ingress filtering is not self-complete. SYN-dog [23] is not self-complete by a similar analysis. It can be shown that the IP traceback systems [19,2,20,27] are also not self-complete.

When an AS deploys a system that is not self-complete, it essentially takes a *good-citizen* strategy to help in a global effort for defeating a certain network threat. But the benefits for itself arrive only after other organizations on the Internet are also good citizens and, moreover, implementing the same defense. In contrast, if an AS joins a self-complete system such as the PAS, it immediately receives the full defense function for traffic between itself and other ASes that also deployed the system. This has a significant practical impact: the immediate benefit during incremental deployment gives incentive for ASes to deploy such a system.

An AS is *PAS aware* if it deploys a PAS on all its BGP border routers; otherwise, it is *PAS unaware*. We are not concerned with the uniqueness for the address of a path whose source or destination is PAS unaware. However, for our scheme to be self-complete, we must ensure that the path address from a PAS-aware source AS to a PAS-aware destination AS (denoted as x) must have a negligibly small probability to be the same as the address from another source AS to x, regardless of whether that source AS is PAS aware or not. This is generally true, as illustrated in Fig. 6, where black circles are routers supporting the PAS and white circles are routers not supporting the PAS. The path address from $AS3$ to $AS1$ is $R3.paddr(AS1) = R3.loc \oplus R1.loc$. It is different from the path addresses from other domains to $AS1$, with one exception: the address from the PAS-*unaware* $AS6$ to $AS1$ is also $R3.loc \oplus R1.loc$. To solve this problem, when $R3$ receives a packet from an external interface (connecting $AS6$) with the P flag being 0, it will set the paddr field to be $R3.paddr(AS1) \oplus R3.loc \oplus r = R1.loc \oplus r$, where r is a random number associated with the external interface. It sets the verification field to be the paddr field XORed with r, which ensures that the verification process will be successful down the path. When the destination $AS1$ receives packets from $AS6$, it will see path address $R1.loc \oplus r$, instead of $R3.loc \oplus R1.loc$.

There is one additional operation. Before a router forwards a packet destined to another AS, if the router's path address to the destination AS is equal to its local number, the router knows that it is the last AS-aware router on the path. In this case, it should disguise the paddr field by hashing before forwarding the packet.

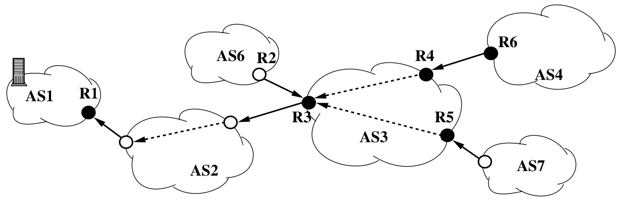

Fig. 6. The path address between $AS3$ and $AS1$ should be artificially made different from the address between $AS6$ and $AS1$.

4. Evaluation

We evaluate our path address scheme both analytically and by simulations, and compare our scheme with the most related work, Pi [27]. The results demonstrate that the PAS is able to drive both the false positive ratio and the false negative ratio to almost zero, while Pi cannot.

4.1. Analysis

Our analytical results reveal some interesting properties of the PAS and Pi.

4.1.1. Analytical model

We first consider an attack involving one malicious host. Suppose that the malicious host has launched an attack, the intrusion detection system at the victim has identified the attack, and it has extracted a path address (if the PAS is deployed) or a path identifier (if the Pi scheme is deployed) from the attack packet. Hoping to block the malicious host, the victim decides to filter all packets carrying the path address (or path identifier). Using this scenario, we try to quantify the false-positive probability and the false-negative probability of the PAS (and Pi).

When a normal host sends a legitimate packet to the victim, if the packet is mistakenly filtered, we call the event a *false positive*. The probability for that to happen is called the *false-positive probability*. Clearly, it is equal to the percentage of all legitimate packets that are filtered, traditionally called the *false-positive ratio*.

When the malicious host launches a new attack, if the attack packet is not filtered, we call the event a *false negative*. The probability for that to happen is called the *false-negative probability*. It is equal to the percentage of all attack packets that are not filtered, traditionally called the *false-negative ratio*.

Let h be the number of routers that have been upgraded to support the PAS (or Pi) on the path from the malicious host to the victim. Let m be the number of bits used to store the path identifier in Pi or the paddr/verification fields and the P flag in the PAS. For Pi, let n be the number of bits in any mark inserted to the path identifier by a router. For the PAS, let p be the number of bits in the paddr field and v the number of bits in the verification field. $p + v = m - 1$. If v is chosen smaller than p, then only v bits in the paddr field are verified.

4.1.2. False-positive probability and false-negative probability of the PAS

Suppose that the victim is blocking the path address extracted from a previously identified packet from the malicious host. Consider a legitimate packet from a normal host. Each router contributes a full p-bit local number to the path address. As long as the routing path from the normal host to the victim has one router that is not in the path from the malicious host to the victim, the addresses of the two paths may differ in any of the p bits. Hence, the chance for these two path addresses to be the same, i.e., the false-positive probability, is

$$FP_{pas}(p) = \frac{1}{2^p}. \quad (2)$$

Next, consider an attack packet from the malicious host. To produce a false negative, the malicious host has to falsify the values in the paddr/verification fields and set the P flag to be 1. The chance for a random value in the verification field to pass the verification, i.e., the false-negative probability, is

$$FN_{pas}(v) = \frac{1}{2^v}. \tag{3}$$

Because $p + v = m - 1$, if m is fixed, we can tune the values of p and v to make a tradeoff between the false-positive probability and the false-negative probability. However, we are able to lower both probabilities if m can be increased. If m is large enough (e.g., 30), we can lower both to almost zero. Finally, by (2)–(3), neither the false-positive probability nor the false-negative probability depends on the distance h from the malicious host to the normal host.

4.1.3. False-positive probability and false-negative probability of Pi

Suppose that the victim is blocking the path identifier extracted from a previously identified packet from the malicious host. Consider a legitimate packet from a normal host. Suppose that the routing path from the normal host to the victim shares the last c hops with the routing path from the malicious host. The path identifier carried in the legitimate packet must share $c \times n$ common bits with the blocked identifier. The packet will be mistakenly filtered if the other $(m - c \times n)$ bits happen to also have the same value as the blocked identifier. Hence, the false-positive probability is

$$FP_{pi}(m, n, c) = \begin{cases} \frac{1}{2^{m-n\times c}} & \text{if } n \times c < m \\ 1 & \text{if } n \times c \geq m. \end{cases} \tag{4}$$

Next, consider an attack packet from the malicious host. This new attack packet follows the same path as the previous one. Hence, the path identifier carried in the packet shares $h \times n$ common bits with the blocked identifier. The other $(m - h \times n)$ bits are randomly set by the malicious host. Hence, the false-negative probability is

$$FN_{pi}(m, n, h) = \begin{cases} 1 - \frac{1}{2^{m-n\times h}} & \text{if } n \times h < m \\ 0 & \text{if } n \times h \geq m. \end{cases} \tag{5}$$

Pi has two design parameters, m and n. Eqs. (4)–(5) reveal some interesting properties of Pi. First, we show a counterintuitive result. It is not true that a larger size for path identifier always leads to better performance. From (4)–(5), increasing m reduces the false-positive probability, but increases the false-negative probability when $n \times h < m$. Decreasing m has the opposite effect. Second, increasing n reduces the false-negative probability, but increases the false-positive probability. Decreasing n has the opposite effect. Third, the false-negative probability increases exponentially as the distance h from the malicious host to the victim decreases. We can tune the values of m and n to make a tradeoff between the false-positive probability and the false-negative probability, but we cannot simultaneously drive them down.

4.1.4. Multiple malicious hosts

The analytical model can be extended to include k malicious hosts. Suppose that the victim is blocking a path address (or path identifier) from each identified malicious host. Through a similar analysis, we have the following false-positive and false-negative probabilities (with superscript k),

$$FP^k_{pas}(p) = 1 - \left(1 - \frac{1}{2^p}\right)^k,$$

$$FN^k_{pas}(v) = 1 - \left(1 - \frac{1}{2^v}\right)^k$$

$$FP^k_{pi}(m, n, c) = 1 - \prod_{i=1}^{k}\left(1 - \max\left\{\frac{1}{2^{m-n\times c_i}}, 1\right\}\right) \tag{6}$$

$$FN^k_{pi}(m, n, h) = 1 - \prod_{i=1}^{k}\left(1 - \min\left\{1 - \frac{1}{2^{m-n\times h_i}}, 0\right\}\right),$$

where c_i is the number of PAS-aware/Pi-aware routers that the routing path of a given legitimate packet shares with the path of the ith malicious host and h_i is the number of PAS-aware/Pi-aware routers on the path from the ith malicious host to the victim.

4.2. Simulations

We use simulations to evaluate the PAS and compare it with Pi in terms of false-positive ratio and false-negative ratio.

4.2.1. Simulation setup

The simulation network has 10,000 nodes (ASes). The nodes are interconnected through their gateways (interdomain routers). A node is said to be *normal* if it does not have an attack host; it is *malicious* if it does. A certain number of nodes are randomly selected to be malicious. The *attacker ratio* is defined as the number of malicious nodes divided by the total number of nodes (which is 10,000). The default attacker ratio is 0.1, but we will vary it in the simulation. A single victim is randomly selected from the network. The network topology is generated based on the Power-Law Internet model [7].

The simulations run on a Window 7 Professional machine of Intel(R) Core(TM) i7 CPU 860 2.80 GHz. For each simulation, the network topology is built first, and some nodes are randomly chosen to be attackers. We implement both Pi and the PAS, which run on the same environment for a fair comparison. We repeat each simulation 1000 times and average the results (false-positive ratios and false-negative ratios, to be elaborated shortly).

The attack model used in our simulations is similar to that in [27]. There are two phases. The first is called the *learning phase*, and the second is called the *attack phase*. In the learning phase, we assume that an intrusion detection system identifies the attack packets and extracts the path identifiers or path addresses (e.g., the most-frequently received ones under a DoS attack) for blocking. How to design an intrusion detection system in general is beyond the scope of this paper. Suppose that, after this phase, the victim learns the path address from each malicious node to the victim, or, if Pi is used, it learns up to r path identifiers for each malicious node. In Fig. 1(c), we have shown that there can be multiple path identifiers from an attacker if it resides near the victim. The default value for r is 1, but we will vary it in the simulation.

In the attack phase, the victim filters all packets carrying the path addresses or path identifiers learned in the previous phase. For the PAS, the malicious nodes generate attack packets with random initial values in the paddr/verification fields and 1 for the P flag. For Pi, the malicious nodes generate attack packets with random initial values in the path-identifier field. We measure the *false-positive ratio*, which is the fraction of legitimate packets from all sources that are mistakenly filtered, and the *false-negative ratio*, which is the fraction of attack packets that are not filtered in this phase.

Pi uses no more than 30 bits in the packet header for its path identifier, including 16 bits from the IP identification field, 1 bit from the flag field, and 13 bits from the offset field. For fair comparison, we allow the same number of bits for the PAS in our simulations. The PAS uses 16 bits for the paddr field, 1 bit for the P flag, and 13 for the verification field. Pi uses all 30 bits for the path identifier. We have learned from Section 4.1 that, in Pi, the number n of bits for a mark represents a tradeoff between the false-positive ratio and the false negative ratio. We let n be 2, 3, 5, or 6 in the

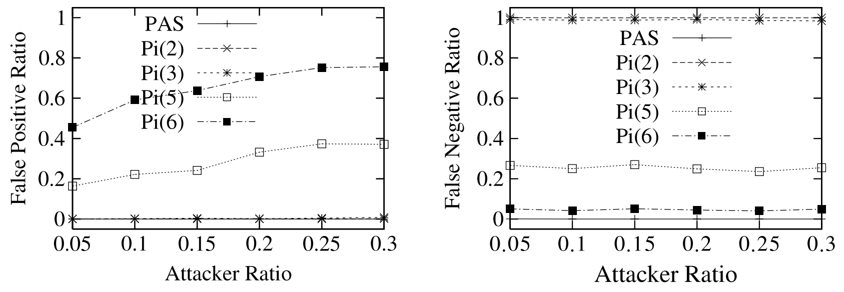

Fig. 7. *Left:* false-positive ratios with respect to attacker ratio. *Right:* false-negative ratios with respect to attacker ratio.

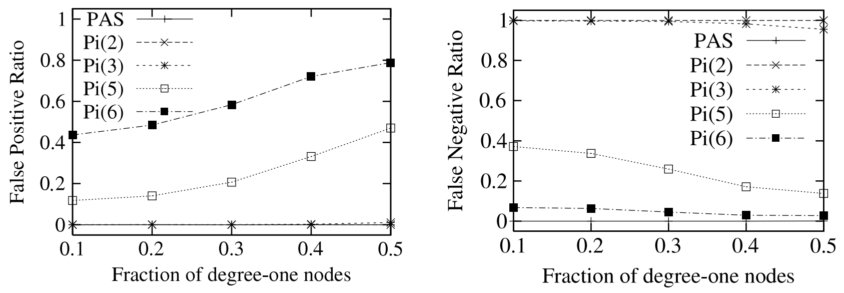

Fig. 8. *Left:* false-positive ratios with respect to the fraction of degree-one nodes. *Right:* false-negative ratios with respect to the fraction of degree-one nodes.

simulations. Simulations using Pi with these n values are denoted as Pi(2), Pi(3), Pi(5), and Pi(6), respectively.

4.2.2. *Performance evaluation with respect to attacker ratio*

In the first simulation, we vary the attacker ratio from 0.05 to 0.3. This means that the number of malicious nodes ranges from 500 to 3000. Fig. 7 shows the false-positive ratios and the false-negative ratios, respectively. Both the false-positive ratio and the false-negative ratio of the PAS are near zero, varying between 0 and 0.0002. The attacker ratio has hardly any impact on the PAS. None of Pi(2)–Pi(6) has a low false-positive ratio and a low false-negative ratio at the same time. The false-positive ratios of Pi(2) and Pi(3) are close to zero, but their false-negative ratios are almost 1. Note that the simulation in [27] did not include malicious nodes close to the victim, but the simulation in this paper does, which reveals the above serious performance problem. Fig. 7 also confirms our analytical result in Section 4.1 that increasing n reduces the false-negative ratio of Pi but increases the false-positive ratio. For the same value of n, as the attacker ratio increases, the false-positive ratio increases while the false-negative ratio decreases. The reason is that, the more the attackers, the more the number of blocked path identifiers (identified in the learning phase), the higher the chance of misblocking normal packets, and the lower the chance of not blocking attack packets.

4.2.3. *Performance evaluation with respect to network topology*

In the second simulation, we study the impact of network topology on the performance of the PAS and Pi. We change the density in network connectivity by varying the fraction of degree-one nodes from 0.1 to 0.5. Fig. 8 shows the false-positive ratios and the false-negative ratios, respectively. The PAS has very small false-positive and false-negative ratios, ranging between 0.0001 and 0.0002. Topology variation has little impact on its performance. For Pi, however, as the fraction of degree-one nodes increases, the false-positive ratios increase while the false-negative ratios decrease. The reason is that increasing degree-one nodes reduces the number of links in the network, which has two consequences. First, it increases the lengths (denoted as c) of common subpaths shared by attack packets and normal packets, and hence increases the false-positive ratio due to (4) in Section 4.1. Second, it increases the path length (denoted as h) and thus increases the false-negative ratio due to (5).

4.2.4. *Performance comparison with respect to r*

In the third simulation, we vary r from 1 to 101. Recall that r is the maximum number of path identifiers per malicious node learned in the first phase. Fig. 9 shows the false-positive ratios and the false-negative ratios, respectively. It is a parameter for Pi, and thus has no impact on the PAS. The larger the value of r, the larger the number of blocked path identifiers in the second phase, the higher the chance of misblocking normal packets, and the lower the chance of not blocking attack packets. Therefore, as r increases, the false-positive ratio of Pi increases and the false-negative ratio decreases.

4.2.5. *Performance evaluation under incremental deployment*

Let the deployment ratio be the fraction of all routers that support the PAS (or Pi). In the last simulation, we vary the deployment ratio from 0.1 to 1.0, and randomly select ASes to be PAS-aware/Pi-aware. Fig. 10 shows the false-positive ratios and the false-negative ratios during incremental deployment. The false-negative ratio of the PAS is always near zero, which means that most attack packets are filtered. The false-positive ratio of the PAS can be high when the deployment ratio is small. The blocked legitimate packets are mostly from PAS-unaware ASes. *The false-positive ratio for PAS-aware ASes is zero*, which is the curve under legend "PAS-aware". This result confirms the PAS's self-completeness in Section 3.6. In comparison, the false-negative ratio of Pi is very high when the deployment ratio is small. It can be surprising that Pi's false-positive ratio may increase when the deployment ratio becomes large. The reason is illustrated by Fig. 1(a) and explained in Section 2—a malicious host behind a long

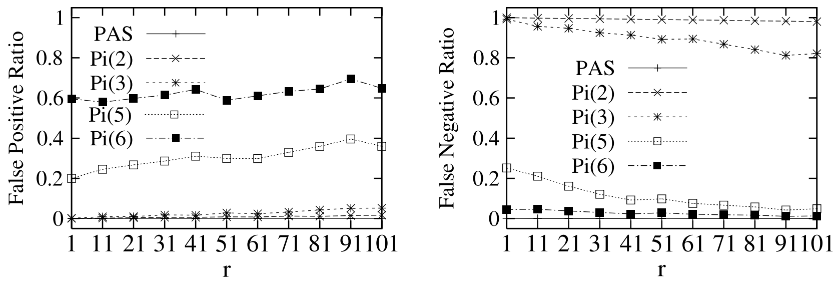

Fig. 9. *Left:* false-positive ratios with respect to *r*. *Right:* false-negative ratios with respect to *r*.

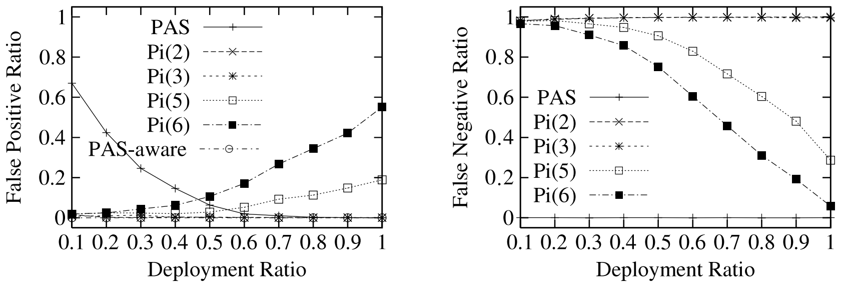

Fig. 10. *Left:* false-positive ratios with respect to the deployment ratio. *Right:* false-negative ratios with respect to the deployment ratio.

routing path, on which many routers implement Pi, causes false positives. A final note is that, unlike the PAS, Pi's false-positive ratio does not change when we only consider Pi-aware ASes.

We also run the above simulations on shifted XOR. Its performance is very close but slightly worse than that of the PAS. We do not plot it in the figures because it almost completely overlaps with the curve of the PAS except for the left plot of Fig. 9, where the false-positive ratio of shifted XOR would be half a percentage higher than that of the PAS.

5. Conclusion

In this paper, we have proposed a novel path address scheme (PAS) that, in an abstract and compact way, makes topological/routing information about the Internet core available at the network edge to assist the development of network security systems. We discuss how to construct path addresses, how to verify if path addresses are authentic, and how to enhance the Internet protocols to support path addresses. Our analysis and simulations demonstrate that the PAS can simultaneously keep the false-positive ratio and false-negative ratio to almost zero.

Acknowledgments

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2011-0004643), and was also supported in part by the US National Science Foundation under grant CNS-1115548.

References

[1] D.G. Andersen, Mayday: distributed filtering for Internet services, in: Proceedings of 4th USENIX Symposium on Internet Technologies and Systems, March 2003.
[2] S.M. Bellovin, ICMP traceback messages, Internet Draft: draft-bellovin-itrace-00.txt, March 2000.
[3] D.J. Bernstein, SYN cookies, http://cr.yp.to/syncookies.html, 1997.
[4] A. Bremler-Barr, H. Levy, Spoofing prevention method, in: Proceedings of IEEE INFOCOM, March 2005.
[5] E.G. de Santerre, S. Jammoul, L. Toutain, Solving the ingress filtering issue in an IPv6 multihomed home network, in: Proceedings of International Conference on Networks, 2010.
[6] A. El-Atawy, E. Al-Shaer, T. Tran, R. Boutaba, Adaptive early packet filtering for defending firewalls against DoS attacks, in: Proceedings of IEEE INFOCOM, 2009.
[7] M. Faloutsos, P. Faloutsos, C. Faloutsos, On power-law relationships of the Internet topology, in: Proceedings of ACM SIGCOMM'99, 1999.
[8] P. Ferguson, D. Senie, Network ingress filtering: defeating denial of service attacks which employ IP source address spoofing, in: IETF, RFC 2827, May 2000.
[9] M. Geva, A. Herzberg, QoSoDoS: if you can't beat them, join them!, in: Proceedings of IEEE INFOCOM, April 2011.
[10] A. Juel, J. Brainard, Client puzzles: a cryptographic countermeasure against connection depletion attacks, in: Proceedings of Network and Distributed System Security Symposium, NDSS'99, February 1999.
[11] A.D. Keromytis, V. Misra, D. Rubenstein, SOS: secure overlay services, in: Proceedings of ACM SIGCOMM'02, August 2002.
[12] T. Li, G. Huston, BGP stability improvements, IETF Internet-Domain Routing, Internet-Draft, draft-li-bgp-stability-01, June 2007.
[13] P. Mahajan, S.M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, S. Shenker, Controlling high bandwidth aggregates in the network, Computer Communications Review 32 (3) (2002) 62–73.
[14] J. Mirkovic, E. Kissel, Comparative evaluation of spoofing defenses, IEEE Transactions on Dependable and Secure Computing 8 (2) (2011) 212–232.
[15] W.G. Morein, A. Stavrou, D.L. Cook, A.D. Keromytis, V. Misra, D. Rubenstein, Using graphic Turing tests to counter automated DDoS attacks against web servers, in: Proceedings of the 10th ACM International Conference on Computer and Communications Security, CCS, October 2003.
[16] M. Moreira, R. Laufer, N. Fernandes, O. Durate, A stateless traceback technique for identifying the origin of attacks from a single packet, in: Proceedings of IEEE ICC, June 2011.
[17] K. Park, H. Lee, On the effectiveness of route-based packet filtering for distributed DoS attack prevention in power-law internets, in: Proceedings of ACM SIGCOMM'01, August 2001.
[18] Y. Rekhter, T. Li, S. Hares, A border gateway protocol 4 (BGP-4), IETF Network Working Group, RFC 4271, January 2006.
[19] S. Savage, D. Wetherall, A. Karlin, T. Anderson, Practical network support for IP traceback, in: Proceedings of ACM SIGCOMM'00, August 2000.
[20] A.C. Snoren, C. Partridge, L.A. Sanchez, C.E. Jones, F. Tchakountio, S.T. Kent, W.T. Strayer, Hash-based IP traceback, in: Proceedings of ACM SIGCOMM'01, August 2001.
[21] C. Villamizar, R. Chandra, R. Govindan, BGP route flap damping, IETF Network Working Group, RFC 2439, November 1998.
[22] X. Wang, M.K. Reiter, Defending against denial-of-service attacks with puzzle auctions, 2003 IEEE Symposium on Security and Privacy, May 2003.
[23] H. Wang, D. Zhang, K.G. Shin, SYN-dog: sniffing SYN flooding sources, in: Proceedings of 22 nd International Conference on Distributed Computing Systems (ICDCS'02), July 2002.
[24] Y. Xiang, K. Li, W. Zhou, Low-rate DDoS attacks detection and traceback by using new information metrics, IEEE Transactions on Information Forensics and Security 6 (2) (2011) 426–437.
[25] A. Yaar, A. Perrig, D. Song, FIT: fast Internet traceback, in: Proceedings of IEEE INFOCOM, March 2005.
[26] A. Yaar, A. Perrig, D. Song, StackPi: new packet marking and filtering mechanisms for DDoS and IP spoofing defense, IEEE Journal on Selected Areas in Communications 24 (10) (2006).

[27] A. Yaar, A. Perrig, D. Song, Pi: a path identification mechanism to defend against DDoS attacks, in: IEEE Symposium on Security and Privacy, May 2003.
[28] S. Yu, W. Zhou, R. Doss, W. Jia, Traceback of DDoS attacks using entropy variations, IEEE Transactions on Parallel and Distributed Systems 22 (3) (2011) 412–425.

MyungKeun Yoon is an assistant professor in the Department of Computer Engineering at Kookmin University, Korea. He received his BS and MS degrees in computer science from Yonsei University, Korea, in 1996 and 1998, respectively. He received his PhD degree in computer engineering from the University of Florida in 2008. He worked for the Korea Financial Telecommunications and Clearings Institute from 1998 to 2010. His research interests include computer and network security, network algorithms, and mobile networks.

Shigang Chen is an associate professor in the Department of Computer and Information Science and Engineering at University of Florida. He received his BS degree in computer science from the University of Science and Technology of China in 1993. He received his MS and PhD degrees in computer science from University of Illinois at Urbana-Champaign in 1996 and 1999, respectively. After graduation, he worked with Cisco Systems for three years before joining the University of Florida in 2002. His research interests include network security and wireless networks. He received the IEEE Communications Society Best Tutorial Paper Award in 1999 and the NSF CAREER Award in 2007. He has been a guest editor for ACM/Baltzer Journal of Wireless Networks (WINET) and IEEE Transactions on Vehicle Technologies. He served as a TPC co-chair for the Computer and Network Security Symposium of IEEE IWCCC 2006, a vice TPC chair for IEEE MASS 2005, a vice general chair for QShine 2005, a TPC co-chair for QShine 2004, and a TPC member for many conferences including IEEE ICNP, IEEE INFOCOM, IEEE ICC, and IEEE Globecom.