Accountable Internet Protocol (AIP)

David G. Andersen[1], Hari Balakrishnan[2], Nick Feamster[3],
Teemu Koponen[4], Daekyeong Moon[5], and Scott Shenker[5]
[1] Carnegie Mellon University, [2] MIT, [3] Georgia Tech, [4] ICSI & HIIT, [5] University of California, Berkeley

ABSTRACT

This paper presents AIP (Accountable Internet Protocol), a network architecture that provides accountability as a first-order property. AIP uses a hierarchy of self-certifying addresses, in which each component is derived from the public key of the corresponding entity. We discuss how AIP enables simple solutions to source spoofing, denial-of-service, route hijacking, and route forgery. We also discuss how AIP's design meets the challenges of scaling, key management, and traffic engineering.

Categories and Subject Descriptors

C.2.6 [**Internetworking**]; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design

General Terms

Design, Security

Keywords

Internet architecture, accountability, address, security, scalability

1 Introduction

We begin by belaboring, with a short list of examples, the trite but true observation that the Internet is rife with vulnerabilities at the IP layer. As amply demonstrated by recent events [7, 27, 37], even a single misconfigured router can wreak widespread havoc on packet delivery. Hijacked routes are routinely used to send untraceable spam [32]. Denial-of-service attacks are so commonplace that they hardly make the news any more. Malicious or compromised hosts spoof their source addresses with impunity, because there is little chance of their being detected.

There is no shortage of proposed fixes to these well-known problems. These solutions, however, often come with one or more of the following problematic requirements:

- *Complicated mechanisms*: e.g., the "capabilities" approach to denial-of-service involves fairly intricate mechanisms that fundamentally change the free-access model of the Internet.
- *External sources of trust*: e.g., S-BGP [19] and similar approaches to BGP security require a trusted certificate authority and a trusted address registry.
- *Operator vigilance*: e.g., using filtering to prevent spoofing requires network operators to keep filters properly configured.

The fact that addressing core vulnerabilities requires significant additional mechanism, external support, or both, suggests that perhaps we are trying to build castles on quicksand. That is, the problem lies not with these proposals in themselves, which represent the best our field has to offer, but with the foundation upon which they were built. In this paper we ask: *what changes to the architecture would provide a firmer foundation for IP-layer security?*

We believe that many of the vulnerabilities listed above are due to the lack of *accountability*: the Internet architecture has no fundamental ability to associate an action with the responsible entity. Real-world security depends on accountability (imagine, if you will, a world where all actions could be taken anonymously), and we think the same applies to the Internet. We thus propose the Accountable Internet Protocol (AIP) as a replacement for the current IP. Our proposal retains the simplicity of the current Internet; in fact, our addressing structure (two or more levels of flat addressing) is much closer to the Internet's original incarnation than today's CIDR-based reliance on aggregation. Where our proposal differs from both the current and past Internet is our use of *self-certifying* addresses for both domains and hosts. This approach, which we first proposed in a position paper [2], allows hosts and domains to prove they have the address they claim to have *without relying on any global trusted authority*. We present the basic AIP design in Section 2. In Section 3, we show how this foundation enables us to deal with the problems of source spoofing, route spoofing, and denial-of-service (DoS) without extensive additional mechanisms, external sources of trust, or extreme operator vigilance.

The AIP approach is not without its challenges. Significant concern has been expressed in the IRTF and elsewhere about the scalability of the current addressing structure [28]. AIP appears to make the problem worse, in that its reliance on flat addresses makes CIDR-like aggregation impossible. In Section 4 we argue that AIP poses no threat to the long-term scalability of the Internet. It may be true that AIP could not be deployed on the current router infrastructure, but here we are more concerned with long-term technology trends than short-term infrastructure realities. We realize that our carefree attitude towards scaling is likely to be controversial, but we hope it represents the beginning of a dialogue on this matter.

Any design that relies heavily on public key cryptography must provide mechanisms to protect against, detect, and deal with key compromise. It turns out that the most subtle issue here is how hosts and domains can detect the presence of an imposter, and we describe this problem and our solution in Section 5.

Finally, any change to addressing must be amenable to traffic engineering. We describe in Section 6 how AIP provides operators (of both transit ISPs and stub networks alike) sufficient tools to accomplish their traffic engineering goals.

SIGCOMM'08, August 17–22, 2008, Seattle, Washington, USA.

2 AIP Design

This section describes the salient features of AIP, starting with the structure of AIP addresses. We then discuss how AIP interacts with the rest of the internetwork architecture, including forwarding and routing, end-to-end (TCP) connections, and DNS.

2.1 Basic Structure and Function

AIP eschews the use of prefixes and CIDR-style addresses, returning to a hierarchical addressing format with two or more components. AIP may thus be viewed as a simple generalization of the Internet's original two-level hierarchical addressing structure where each address had a network and a host component and routers inspected only the (flat) network portion until the packet reached the destination network.

Unfortunately, addressing has become more complicated with the advent of autonomous systems (used in BGP routing) and classless routing (CIDR), with no clean mechanism to map autonomous systems to prefixes. To redress this shortcoming, AIP removes the distinction between an autonomous system identifier and the set of routes (prefixes) it can advertise, using the same handle to name both.

The AIP design assumes there are some number of independently administered networks (as is the case today), operated by distinct administrative units. Each administrative unit decomposes its network into one or more *accountability domains* (ADs), each with a unique identifier. Each host is also assigned a globally unique endpoint identifier (EID). Analogous to the original Internet addressing structure, the AIP address of a host currently homed in some AD would have an address of the form `AD:EID`.

To handle the case of a host that attaches multiple times to the same AD (e.g., with both a wireless and a wired Ethernet connection), the final eight bits of the EID are *interface bits* that give each interface a unique identifier: EIDif_1, EIDif_2, etc.

Each AD is visible in the wide-area routing protocol, so one might think of each AD as corresponding to a BGP prefix in the current Internet. Some ADs might be quite large, preferring to organize themselves hierarchically internally. To support this requirement, AIP supports multiple levels in the hierarchy, so in general an AIP address would have the form $\texttt{AD}_1\texttt{:AD}_2\texttt{:...:AD}_k\texttt{:EID}$.

Eliminating structure in the `AD` and `EID` allows us to make them *self-certifying* [26]. The notion of a self-certifying name is straightforward: the name of an object *is* the public key (or, for convenience, the hash of the public key) that corresponds to that object. In AIP, the `AD` is the hash of the public key of the domain, while the `EID` is the hash of the public key of the corresponding host. Although higher layers have used self-certifying naming (e.g., hosts, data, and services) [26, 43], and HIP [29] uses such addresses in a shim layer between the IP and transport layer, AIP is the first architecture to our knowledge that uses fully self-certifying addresses at the internetwork layer itself. One result of self-certification is that each hierarchical component in an AIP address is 160 bits long (Figure 1).

Our use of self-certifying addresses follows from a simple line of reasoning. Accountability requires a verifiable identity, and in a network setting the only practical method of verification uses cryptographic signatures. To use such signatures, identifiers must be bound to their public key. Security, however, should not rely on extensive manual configuration or globally trusted authorities, so the keys must be intrinsic to the identifiers. Thus, we believe that self-certification is an indispensable aspect of providing accountability at the network layer.

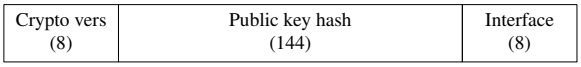


Figure 1: The structure of an AIP address. For AD addresses, the interface bits are set to zero.

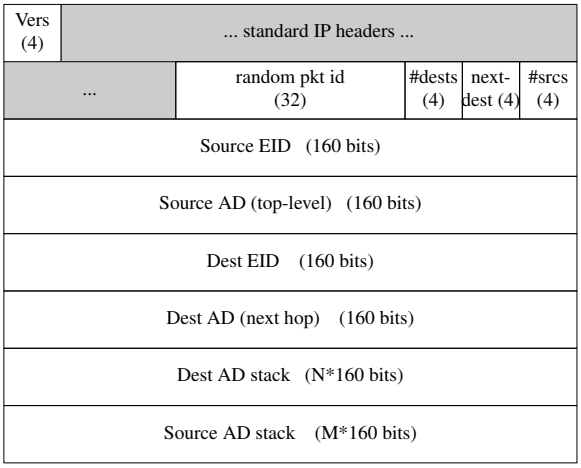


Figure 2: The AIP packet header.

Existing schemes (e.g., S-BGP [19]) implement this binding between identifiers and their public keys using registries that map identifiers to their public keys (a PKI). Unfortunately, these registries must be both up-to-date (via manual configuration) and globally trusted. Unfortunately, experience with Internet address registries suggests that one cannot rely on manual configuration to keep registries accurate and up-to-date [13, 16, 36]. Self-certifying addressing frees security mechanisms from undesirable trust relationships or manual configuration. Existing IP and transport security mechanisms (e.g., IPsec [18]) could also use AIP's self-certifying address structure to securely establish the identity of a remote host without relying on an external infrastructure.

Because AIP uses cryptographic primitives whose strength may degrade over time, each AIP address (Figure 1) contains a version number that indicates what signature scheme incarnation was used to generate the address. In Section 5, we discuss how this field may be used to accommodate the gradual evolution of the digital signatures used in AIP to cope with the (inevitable) weakening of earlier schemes.

2.2 Forwarding and Routing

Packets contain the destination's `AD:EID`, as shown in Figure 2. Until the packet reaches the destination AD, routers use only the destination `AD` to forward the packet. Upon reaching the destination `AD`, routers forward the packet using only its `EID`.

Forwarding to a destination that has more than one `AD` proceeds identically to reach the first `AD` in the destination AD stack: intermediate domains examine only the next hop destination AD. The destination AD's border router examines the additional destination ADs fields and replaces the next hop destination AD with that pointed to in the destination ADs stack by the next-dest field, and increments the next-dest pointer. Most routers therefore examine only the dest

AD field (placing the burden of hierarchical routing only on those domains with an internal hierarchy), but the entire destination stack is preserved for the recipient to examine.

Interdomain routing: In AIP, interdomain routing occurs in much the same way that it does today (and can benefit from any future improvements to BGP or use a different inter-domain routing protocol). Today's Internet uses prefixes as the routing objects; in contrast, AIP's routing objects are AD identifiers, so interdomain routing occurs entirely at the AD granularity. BGP advertisements are for the ADs. Routers in an AD maintain routing information on a per-AD basis; i.e., an AIP routing table maps AD numbers to "next hop" locations but does not maintain any information about EIDs in other ADs. Each router also participates in an interior routing protocol (e.g., OSPF) to maintain routing information to the EIDs within the AD. We expect the internal routing protocols to handle at most a few tens of thousands of flat entries, a capability well in line with modern switches.

Although routing is done on a per-AD basis, path descriptions might be done on a larger granularity. AIP continues to support the notion of an autonomous system (AS) because an organization may not wish to advertise its internal AD structure through its BGP routes for various reasons, including: (1) describing paths on such a fine granularity might increase routing churn; (2) peering with an AS (e.g., AT&T) is simpler than peering with five sub-ASes (AT&T-chicago, AT&T-nyc, etc.); (3) configuring policy may be easier at the granularity of an AS; etc. The path descriptors in BGP could use a separate set of self-certifying identifiers that would identify the organizational entity (such as an AS) but not the smaller AD. These path descriptors are also 160-bit self-certifying AIP addresses, but they have no EIDs contained within them as an AD would—they are used purely for routing to the destination AD.

2.3 DNS and Mobility

The domain name system would include an AIP-record containing the AIP address(es) for a hostname. A host might have multiple addresses if it had direct upstream connectivity to multiple domains AD_i; the host would then have addresses AD_i:EID in its AIP-record for each domain. If a host had multiple interfaces, an entry would appear as AIP addresses returned in the AIP-record.[1]

Mobility support is based on the self-certifying endpoint identifier (EID) part of the addresses and the use of an end-to-end mobility protocol. Transport protocols on top of the AIP layer bind to the source and destination EIDs, which remain unchanged while hosts roam from one AD to another, even though the AD part of the addresses changes. Thus, to keep traffic flowing it is sufficient for a roaming host to instruct its remote hosts to migrate their traffic from an old address to the new one. For that purpose, AIP adopts the mechanisms of TCP Migrate [38] and HIP [29].

For initial rendezvous, mobile AIP hosts maintain their current location (AD) in DNS. AIP's self-certifying structure again simplifies handling dynamic DNS: the DNS server can be configured to allow an EID to update an existing host → AD:EID binding that currently points to that EID. The server and DNS operators do not need to maintain a separate update key to control dynamic updates. The same keying advantages apply to a faster-moving host that wished to use a "home agent" to relay traffic to it.

[1]To achieve the full benefits of AIP, the DNS records would best be served using a secure DNS variant to prevent an attacker from directing clients to alternate destinations by modifying DNS responses. Most of AIP's other advantages, including anti-spoofing, secure BGP, and shut-off techniques, do not depend at all on DNS.

3 Uses of Accountability

Given AIP's basic design, we now describe how this accountability foundation can be used to provide better network-layer security.

3.1 Source Accountability: Detecting & Preventing Source Spoofing

Source address spoofing refers to the problem of a host using a source address that has been assigned to another host. If a source uses a spoofed address at which it cannot receive packets, then higher-layer protocols that use a three-way handshake before instantiating any state or expending computation will not be significantly affected. Not all higher-layer protocols use such a mechanism, so detecting this situation will be useful. A harder-to-detect form of spoofing occurs when a malicious or compromised host uses a source address at which it can to receive packets. Such attacks have been observed in the Internet [39] and are used, e.g., to send spam [33]. These attacks can arise because of spurious route propagation or because the spoofing host is on a shared network (e.g., wireless). We are interested in detecting both forms of spoofing. We are also interested in limiting the damage that can be caused by *address minting*, in which a host can create a large number of distinct (unused) addresses for itself.

One approach to coping with spoofing is to use ingress filters, but the success of this approach has been rather limited in practice [6, 10]. We believe that this lack of success has less to do with the mechanism itself (routers implement it at line rates), but reflects our more general thesis that schemes that depend on correct operator action are often only marginally effective. AIP's source accountability, in contrast, uses self-certifying addresses to develop simple mechanisms that verify the source of packets, dropping the packets if the source addresses are spoofed. Our mechanism requires no configuration or interaction by operators or end-users. Its goal is to prevent spoofing by entities not on the direct path from the source to the destination—a router on-path from A to B could still spoof packets from A, though it could not sign them to prove authenticity.

AIP's source accountability mechanism extends (and renders more widely useful) "unicast reverse path forwarding" (uRPF) [11]. uRPF is an automatic filtering mechanism that accepts packets only if the route to the packet's source address points to the same interface on which the packet arrived. uRPF is currently useful in an edge network to prevent spoofing by single-homed clients, but it cannot cope with multi-homed customers and, because of route asymmetry, it does not work in the core. AIP's source accountability mechanism essentially combines uRPF with a second mechanism to automatically verify if packets are valid even if they arrived on an interface other than the reverse route to the destination.

Recall that the AD and EID components of an address are hashes of public keys. We use these public keys to validate the source address of a packet in two places. First, each first-hop router[2] verifies that its directly-connected hosts are not spoofing. Next, each AD through which a packet passes verifies that the previous hop is a "valid" previous-hop for the specified source address. The process for verifying a packet's source address, AD:EID, summarized in Figure 3, is as follows:

EID verification: If the first-hop router or switch, R, has not recently verified the source host, it drops the packet and sends a *verification packet*, V, to the source. R avoids maintaining state for

[2]By first-hop router, we mean the first router trusted by the network operator of the stub network to which the host in question belongs.

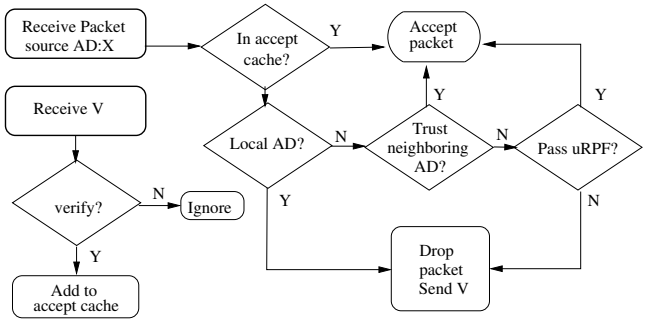


Figure 3: Process for verifying a packet's source address.

Let:

rs	=	Per-router secret, rotated once per minute
$\text{HMAC}_{\text{key}}\langle \text{M} \rangle$	=	Message authentication code of M
$H\langle P \rangle$	=	Hash of P
iface	=	Interface on which packet arrived

Source $S_{AD} : S_{EID} \rightarrow$ **Dest** $D_{AD} : D_{EID}$
Packet P.

Router R1 $\rightarrow$ **Source**:
Verification packet V =
$\text{HMAC}_{rs} \langle S_{AD} : S_{EID} \rightarrow D_{AD} : D_{EID}, H\langle P \rangle, \text{iface} \rangle$

Source $\rightarrow$ **R1**:
$\{\text{accept}, K_{S_{EID}}, V\}_{K^{-1}_{S_{EID}}}$

Figure 4: Source address verification protocol.

each V using the protocol in Figure 4. V contains the source and destination AIP addresses of the original packet, the packet's hash, and an encoded representation of the interface on which the packet arrived. R signs V with a message authentication code (HMAC) using a secret, rs, known only to R, which it rotates periodically (e.g., once per minute). The sender must prove that it has identity `EID` by signing V with the private key associated with `EID`. If the host produces the correct signature, then R caches this information and forwards subsequent packets as well. The host must re-send the packet that generated the verification packet, because R drops all unverified packets. For complete protection, this mechanism might need to be implemented in network switches, or would need to be linked to some switch-level ARP security mechanism.

The verification packet includes the hash of the packet that triggered the exchange. Hosts must not respond affirmatively to verification requests for packets they did not originate. Therefore, each host maintains a small cache of the hashes of very recently sent packets. The random packet ID in the AIP header ensures that each of these hashes is highly likely to be unique.

AD verification: When a packet crosses an AD boundary, the incoming AD must decide if the source address is valid. For a packet entering AD A from AD B, AD A performs the following checks:

1. If A trusts B to have performed the appropriate checks on the packet's source address (as might be the case between pairs of tier-1 or mutually trusted ISPs), then A forwards the packet.
2. If A does not trust B, then A performs uRPF checks to determine whether the packet arrived on the same interface that the return route to its source would take. If uRPF succeeds, A forwards the packet.
3. If these tests fail (e.g., in the case of route asymmetry), A drops the packet and sends a verification packet to `AD:EID` using the same protocol used for `EID` verification in Figure 4. If EID replies affirmatively, the router adds an entry permitting subsequent packets from `AD:EID` to pass when they arrive on the verified interface.

Below, we discuss the properties this mechanism provides and how routers can scalably handle large numbers of flows.

Accept cache management: When a router receives a signed response to a verification packet, it adds an entry to its *accept cache* that permits the passage of subsequent packets from `AD:EID` arriving on interface *iface*. To bound the size of the accept cache, a router with more than a threshold number of entries T for a single `AD` will upgrade the accept cache entry to an `AD`-wildcard accept: `AD:*` and will remove the individual `AD:EID` entries.

This state required by this mechanism scales well with the number of hosts because of the division of filtering responsibility in the network. Routers need not maintain accept cache entries for any ADs that pass uRPF checks, or for sources coming from trusted domains. As a result, we view the tasks of routers in a network as follows:

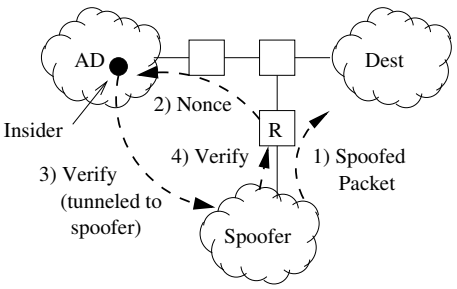
Border routers must verify the source addresses of packets arriving from customers whose return path does not go directly to the customer. Such customers are primarily those who use an ISP or link for backup routing.

Interior routers can trust the verification decisions made by border routers and need not perform any further actions.

Peering routers to large peers will likely be configured to trust the peer's verification based upon a bilateral contractual agreement. Such agreements benefit both parties, reducing their filtering load.

An attacker, then, can only increase the number of accept cache entries by T for each AD he controls. Below, we discuss how ISPs can limit the number of ADs that a malicious customer can create.

Protecting those who protect themselves: The anti-spoofing protocol and accept cache management provides defenses for ADs that protect themselves. The protocol as described admits the following insider attack against a source AD:



1. An attacker outside of `AD` sends a spoofed message:
   $AD : EID_{\text{insider}} \rightarrow \text{Dst}$
2. A router along the path replies with the verification packet and sends it to the colluding insider, $AD : EID_{\text{insider}}$.
3. The insider creates a verification packet, and tunnels it to the spoofer.

4. The spoofer sends the response to the verification packet to *R*, instructing the router to permit the communication.

If there are enough compromised hosts inside AD, they can create enough verification packets to cause a target router to upgrade to a wildcard entry allowing AD:$\star$ from the spoofing domain, and the spoofer will then be able to spoof arbitrary EIDs within AD. This number must exceed the upgrade threshold T. We discuss below how ADs can prevent arbitrary address minting by their nodes.

The effect of this mechanism is that an AD that has few compromised nodes and that does not permit its nodes to spoof a large number of EIDs cannot be spoofed, encouraging good Internet hygiene. An AD with many misbehaving hosts could be spoofed by a sophisticated attacker. It is noteworthy that this mechanism ensures, e.g., that an attacker can only mount a DoS reflector attack against a victim that he has already extensively compromised. Note that such spoofing does *not* affect security-critical traffic; such traffic can still be unforgeably signed and encrypted using the EID's key.

A remedy to this attack is to also require an AD domain signature on the verification packet responses from AD and to have the router verify that the interface on which the verification arrived matches the interface of the packet that initially triggered the verification packet. The verification packets would then be sent from the insider to a designated domain signer inside AD, which would forward the response to R. The cost of this mechanism is that it requires that the path from the source node to the destination match the path from the domain signing node to the router. Whether the increased security is worth the increased complexity is an open question.

Limiting address minting: AIP detects address spoofing, but nothing in the design prohibits a malicious host from creating an arbitrary set of EIDs, or a malicious domain from making up routing announcements for an arbitrary number of fake ADs. Minting could be used for DoS purposes or to circumvent filters that blocked a particular host or domain.

We do not propose an architectural solution to this problem, but rather note that AIP admits a straightforward engineering solution: Because an attacker cannot claim the identity of another, an AD can simply limit the number of new EIDs or ADs that each of its hosts and customers are allowed to announce. The solution is similar in both cases:

EID limiting: The first-hop router or switch places a unique EIDs/second limit on each port. This mechanism already exists in many switches as "Port Security" to guard against MAC spoofing.

AD limiting: Today, operators may or may not configure limits on the set of prefixes and ASes their customers announce, either manually or from databases such as the RADB. Using AIP, operators instead merely limit the number of unique ADs that their customer may announce. Using such a limit gives the customer flexibility about how they run their network (and does not require them to contact their ISP to add a few new ADs), but prevents gross abuse.

3.2 Shut-off Protocol

Although AIP's source accountability directly eliminates some classes of DoS attacks that rely on source address spoofing, other attacks remain unaffected, such as flooding a victim with traffic from compromised hosts. AIP's self-verifying addressing enables a natural way to throttle unwanted traffic, in which a victim host sends an explicit "shut-off" message to a host sending such traffic. This method uses an idea first suggested by Shaw to throttle DoS traffic from "well-intentioned" hosts [35], but AIP enables a considerably simpler and more general solution than the current IP architecture does. The approach is also similar to the AITF mechanism [4], except that we rely on network interface cards (NICs), not gateway filters, and again benefit from the properties of the addressing scheme, to develop a new *shut-off protocol*.

Although the vulnerabilities caused by the complexity of modern software make it difficult for the owners to prevent compromises, they are typically well-intentioned and do not launch attacks of their own volition. We envision having these well-intentioned owners equipping their hosts with a smart network interface card ("smart-NIC") that helps control the network behavior of the host by selectively suppressing or rate-limiting packet transmission.[3]

Protocol: The smart-NIC records the hashes of recently sent packets and accepts a special class of packets called *shut-off packets* (SOPs). A SOP sent from host X to host Y includes a hash of a recent packet sent to X from Y and a time-to-live (TTL), all signed by X:

Zombie $\rightarrow$ **Victim**: Packet P.

Victim $\rightarrow$ **Zombie**:
$\{\text{key} = K_{\text{victim EID}}, \text{TTL}, \text{hash} = H\langle P\rangle\}_{K^{-1}_{\text{victim EID}}}$

Upon receiving an SOP, the smart-NIC first checks to see if it had sent a packet whose hash matches that in the SOP (discussed below). If not, it disregards the SOP; if so, it installs a filter suppressing further packets from Y to X for the duration of the TTL. The mechanism is designed to "fail-open": the TTL prevents permanent shut-off and the filters are stored as soft state.

AIP's combination of self-certifying addresses and spoof prevention makes this approach feasible. X's signature and its key assure Y that X (or at least someone with X's private key) has sent the request. The hash of a recent packet proves that Y has recently sent a packet to X. This proof is necessary to prevent replay attacks, to prevent an attacker from exhausting the filter state in the NIC to allow them to continue attacking a chosen victim, and to ensure that even if an attacker circumvents AIP's anti-spoofing, it cannot cause a remote machine to block communication with the victim.

It is important that the shut-off process not require a three-way handshake because a host under attack may not receive the return packets.

We note that this mechanism will not stop particularly determined attackers. Out-of-band mechanisms will undoubtedly still be needed to cope with them. In Section 5, we discuss how AIP enables self-certifying registry entries about domain ownership, etc., that can facilitate out of band remediation.

Preventing bypassing: The smart-NIC must protect its firmware and configuration by requiring physical access to modify it, e.g., by plugging it into a USB or serial interface. As a result, the shut-off mechanism is unmodifiable from the host. Attackers cannot circumvent the mechanism, as every packet goes through the NIC to reach the host.

Preventing preemptive shut-off: An attacker might begin an attack by sending a shut-off packet to the victim, to prevent the victim from stopping the attack. To prevent this attack, the smart-NIC permits the victim to send a low rate (e.g., 1 packet per 30 seconds) of shut-off packets to an already-blocked host.

Replay prevention: An attacker might try to spoof SOPs to a victim to prevent it from communicating with a legitimate host. Our

[3] Note that we assume well-intentioned owners only for preventing botnet-style DoS attacks; other protocols built using AIP, such as the anti-spoofing protocol, protect against malicious host owners as well.

mechanism safeguards against this attack in two ways. (1) All packets must be signed by the victim's private EID key. The legitimate host would therefore have had to previously send a shut-off packet to the victim. Spoofing a SOP is difficult under AIP because of its address spoofing prevention, but it could happen if the attacker subverted the address spoofing prevention at the victim or at the legitimate host. (2) The SOP must include the hash of a previously sent packet from the victim to the legitimate host. In addition to whatever changing content is in packets (sequence numbers, nonces, etc.), AIP uses its 32-bit random packet ID to ensure that previously sent packets cannot be used as the basis of a SOP replay attack.

A replay attack can only be mounted *when the attacker has already sniffed a legitimate shut-off packet between the two hosts in question*. This requirement already sets a high bar for attempting to abuse SOP packets, and so we are willing to accept a very small false positive rate for replay prevention to reduce memory requirements. The smart-NIC therefore records the transmitted packet hashes using a Bloom Filter, sized as follows:

Packets per second: The majority of current hosts transmit far fewer than 50,000 packets per second. Because the shut-off protocol targets the common case of well-intentioned end-hosts, servers that generate an unusually high volume can simply disable the shut-off protocol on their NICs with little impact: these machines are comparatively few and are typically professionally managed.

False positive rate: The maximum shut-off duration is 5 minutes. We size the false positive rate such that an attacker would have to flood a victim with a high volume of packets for roughly this duration before finding a filter collision. Each shut-off packet is roughly 550 bytes (most of which is a 2048-bit public key and signature). For recency, we require the victim to send a packet received within 30 seconds to reduce the state that the NIC must maintain. We desire for an attacker to have to send 100 Mbits/s of traffic for ≥ 5 minutes (about seven million packets) to cause a 5 minute interruption. We therefore use as Bloom filter parameters:

$$\begin{aligned} n \text{ elements} &= 1,500,000 \\ m \text{ bits in table} &= 64 \times 2^{20} \text{ (8MBytes)} \\ k \text{ hash functions} &= 12 \end{aligned}$$

The probability of a false positive is 2.9×10^{-8}, or about 1 in 35 million. The 12 hash functions can be computed as 26-bit sections of a hash computation such as SHA-384. Note that this false positive rate is quite conservative: the probability is much lower if the source has been (as would be typical) transmitting at less than 50,000 packets per second.

Because this mechanism is designed to defend against high-volume floods, the NIC can simply clear the Bloom filter every 30 seconds. If a victim is unlucky and responds to a packet sent just before the filter is cleared, it can try again a few seconds later.

3.3 Securing BGP

AIP greatly simplifies the task of deploying mechanisms similar to S-BGP [19] to secure the routing system against hijacking and route forgery.

This task is difficult today because IP lacks a firm binding between public keys, autonomous systems, and the prefixes announced in routing messages. As a result, securing BGP requires external trusted registries that bind, e.g., an owner's public key to a prefix or to an AS number. In part due to the difficulty of creating, maintaining, and trusting these registries, the deployment of secure routing protocols has languished despite considerable attention in both the research and operational communities.

AIP eliminates the need for these databases: In AIP, the network an AD (or AS) announces *is* the AD itself, which eliminates the need for key-to-AS registries. Only a router or network in possession of the private key corresponding to `AD` can generate authentic routing messages. As a result, secure routing follows naturally from AIP, using mechanisms nearly identical to S-BGP:

1. Operators configure a BGP peering session. By identifying the peer AD, the session is automatically aware of the public keys that should be used to verify announcements from the peer and to negotiate an encrypted communication session with the peer.
2. BGP routers sign their routing announcements. A router receiving a routing update verifies the signature before applying the changes or forwarding the announcement. We discuss the resource requirements of the resulting load in Section 4.2.2.
3. Each router must be able to find the public key that corresponds to an `AD`. These keys could be transmitted in-line with BGP messages, or could be sent as an out-of-band, slowly changing database, as in S-BGP. As discussed in Section 5, this distribution is quite simple: because ADs are the hash of the corresponding public key, the bindings are completely self-certifying.

4 Routing Scalability with AIP

In this section, we examine the question of whether the combination of AIP and the continued growth of the Internet will cause some aspect of Internet routing, such as the forwarding and routing information base (FIB and RIB) sizes, update rate, etc., to exceed the capabilities of future hardware to support in a cost-effective manner.

We find through our analysis—using even conservative estimates for future hardware capabilities—that neither the continued growth of the Internet nor the introduction of AIP should impose an undue scaling burden. ISPs may still find the effects of routing growth undesirable, because they could be forced to upgrade routers, but our analysis is strictly one of the *possible*: we argue that future routers will be able to support the larger routing table sizes without an increase in price relative to today's routers.

Resources affected by routing growth: We begin by more precisely defining what we mean by "routing growth" and the physical resources that would be affected by such growth:

Growth of	Resource affected
Routing table size	DRAM
FIB size	DRAM/SRAM/CAM
Update processing	CPU

To understand the effect of growth upon these resources, we first examine current Internet growth curves and survey several predictions for future routing table sizes and update rates. We then estimate the effects of moving to AIP-style routing. Using the estimates for routing table size, FIB size, and update rate, we then explore whether semiconductor technology trends will be able to meet these demands at constant cost.[4]

[4] In this analysis, we count "backbone" BGP announcements. Many providers have significantly more prefixes de-aggregated internally or from layer-3 virtual private networks (VPNs). Our focus here is on the *change* in the relative numbers of prefixes more than on the absolute number. An assumption in this analysis, therefore, is that the internal prefixes will scale at the same rate as the external prefixes.

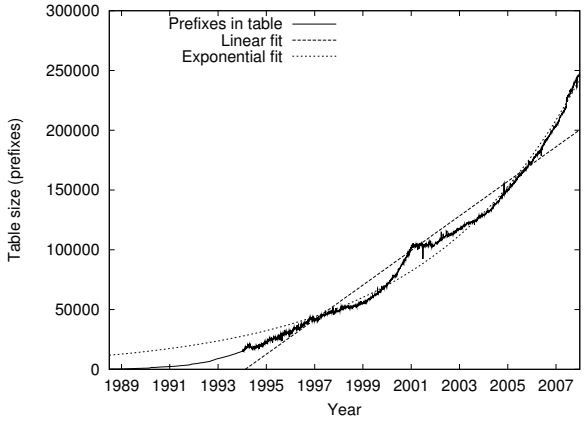


Figure 5: Routing table size growth in prefixes. The exponential fit represents a 17% yearly table growth. The bump in 2001 is most likely due to the dot-com boom and subsequent crash.

Year	17% Growth	Fuller
2008	Observed: 247K	
2011	396K	600K–1M
2020	1.6M	1.3M–2.3M

Table 1: Prediction of table sizes. The first column shows the size if growth continues at 17% per year; the second column reproduces the predictions from Fuller et al.

4.1 Routing Growth Estimates

Diameter of the Internet / AS path length: According to Leskovec et al. [23], the AS-diameter of the Internet has been *shrinking.* From November 1997 to January 2000, the average out-degree of an AS increased from roughly 3.6 to 4.1, and the AS diameter decreased slightly from nearly 4.8 to 4.6 AS hops. Our analysis points to a slightly different conclusion: the average AS path length received at Routeviews in December 2007 was 4.52 entries (2.3 billion announcements) while in December 2001 it was 4.30 (387 million announcements). Both results support the conclusion that, if current trends continue, the increase in the AS-diameter in the future is likely to be small.

Routing table size: Our best estimate for routing table growth is that for the last ten years, the table size has been growing at roughly 17% per year. Figure 5 shows the table growth with both linear and exponential regression lines.[5] The exponential fit is $size = 2.07 \cdot 10^4 \cdot e^{4.253 \cdot 10^{-4} \cdot day}$ with day 0 being June 30, 2008. The 17% growth prediction is compatible with predictions from Fuller et al. [12]; *if* past and current trends continue, the routing table is likely to have about 1.6 million entries by 2020 (Table 1).

Churn: The amount of routing traffic appears to grow *roughly* linearly with the routing table size, but the picture is less clear than the simple table scaling. Figure 6 shows the number of prefixes announced and withdrawn *per week* by the AT&T RouteViews peer. During 2002, the average daily volume was 122,966 updates per day. During 2007, the number was 304,996, an increase of 248%. The routing table grew from 107,424 prefixes (1.145 updates/prefix/day) to 247,167 prefixes (1.234 updates/prefix/day) during the same period, an increase of 230%.[6] The change in churn appears to be affected by more factors than simply growth, as the rapid increase and subsequent correction in 2005–2006 indicate. In the absence of more information, we assume that *per-prefix* churn will remain relatively stable or increase slightly in the coming years, using 1.5 updates per prefix per day as a conservative estimate.

[5] Data from Geoff Huston.

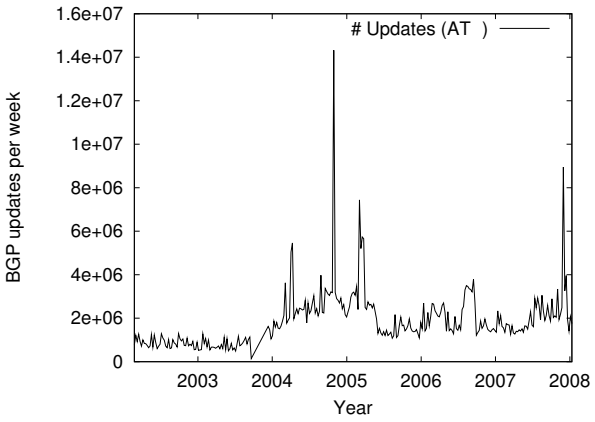


Figure 6: The number of prefixes announced and withdrawn per week by the RouteViews AT&T peer router.

Year	Growth to 1.5 up/prefix	Fuller/Huston
2006/7	305K	700K
2011	593K	-
2012	694K	2.8M
2020	2.4M	-

Table 2: Predictions of total BGP update volume.

Our resulting predictions (Table 2) are smaller than the Fuller/Huston predictions; we believe this difference arises for two reasons. First, one set of Huston's measurements focused mostly on 2005, which appears in retrospect to have shown anomalously large growth in routing volume (in fact, volume *decreased* markedly in the following year). Second, the measurements may differ by a constant factor simply because of the different vantage points—as Table 2 shows, the 2006-2007 measurements themselves differ by a factor of two. Fuller's caveat that these numbers represent a "cloudy crystal ball" (low-confidence projections) holds for our churn analysis as well.

The overall rate of updates with these predictions is small (1.6M daily updates is only 28 updates/sec on average). **The more important update rate is therefore during full table updates.** When a BGP session resets, by 2020, the routers will have to exchange ≥ 1.6 million prefixes with each peer, ideally in a few seconds.

4.2 Effects of Moving to AIP

AIP will have several effects on routing, both positive and negative.

RIB and FIB size increase: The move to AIP will increase the size of the RIB and FIB in two ways. First, the size of prefixes and ASes will increase from their current 32 and 16 bits to 160 bits. Second, just as the move to S-BGP would require, a router will need to store a copy of each AD's public key. For domains that use a two-level key hierarchy (Section 5), this requirement will be

[6] These numbers include session resets to the monitoring node.

Component	2007	2011	2020
DRAM capacity ($Gbits/cm^2$)	1.94	5.82	46.52
DRAM access time (ns)	< 15	< 15	< 15
SRAM capacity ($Gbits/cm^2$)	As transistors		
SRAM access time (ps)	400	-	70(2018)
10^9 Transistors/CPU	1.1	2.2	17.7
DRAM $/Gbit	9.6	2.4	0.3
High-perf CPU $/$10^9$ transistors	122	30	1.3

Table 3: ITRS projections for DRAM, DRAM, and CPU.

doubled. While the number of RIB entries is proportional to the number of peers, each AD's full key must only be stored once. For this discussion, we assume 2048-bit RSA public keys.

An important change with AIP is that FIB lookups become flat. The effects of this change are somewhat difficult to measure, and so we deliberately leave this improvement out of our calculations to ensure a worst-case estimate for AIP. In practice, we expect flat lookups to require roughly 5× fewer memory accesses than prefix-based lookups would using modern DRAM-based lookup algorithms [40].

CPU costs for cryptographic operations: The cryptographic costs of AIP are very similar to those for adopting S-BGP. Routers must verify signatures on incoming routing announcements and sign their own announcements.

Diameter changes: The move to AIP may result in some large domains being split into multiple ADs. We envision, for instance, that a large ISP might split itself into a transit AD and a number of ADs from which it allocates direct customers. As a consequence, the move to AIP might increase the diameter of the network, perhaps by two or three AD-hops. This discussion is, of course, highly speculative, as it depends on the reaction of the operational community to AIP. As a result, we present scaling numbers with the diameter unchanged and with it increased.

4.2.1 Semiconductor Growth Trends

The International Roadmap for Semiconductors (ITRS) is the semiconductor industry's joint technology roadmap for semiconductor technology development through 2020 [1]. Its predictions are an engineering extension of Moore's Law, for the last 20 years brought to fruition by the resultant industry efforts to *meet* those predictions. Whether they will continue to hold through 2020 is uncertain, but past experience suggests that the roadmap is surprisingly accurate. A noteworthy feature of the 2005 and 2006 ITRS updates is that they reduce their doubling predictions from 1.5 years (historical) to 3 years.

In general, the roadmap predicts a continuation of the aggressive growth observed thus far (Table 3). Both DRAM density and CPU performance are expected to improve dramatically. While SRAM access speeds will similarly increase, conventional DRAM access and write times are expected to improve only modestly, unless a shift to faster memory technologies such as MRAM occurs.

4.2.2 Resource Requirements

RIB storage (DRAM): The RIB must hold one copy of the routing table from each peer, and requires that the DRAM be able to sustain enough throughput to load this data from peers at high speeds.

System	2007	2011	2020
IP	0.386 ($30)	0.711 ($14)	2.9 ($7)
AIP	1.0 ($81)	1.7 ($34)	7 ($17)
AIP-Diam	1.3 ($103)	2.0 ($40)	8.2 ($21)

Table 4: RIB memory requirements projections in GBytes, and the projected *production* costs of that memory. The actual cost projections ignore scaling factors due to low volume or high speed requirements, but the cost trend should apply to general purpose as well as specialized memory. AIP-Diam shows the requirements if AIP causes a 60% increase in the diameter of the Internet.

In the worst case, the RIB is stored as separate entries for each IP prefix (today) or AD (in AIP), with no compression of shared AS paths. Today, each AS takes 16 bits; in the future, each AS will likely require 32 bits [42]. With the diameter of the Internet not increasing substantially, we assume the average AS path length will be 6 (today it is slightly under 5). We conservatively assume that each router may have up to 20 peers sending it a full table. As discussed above, moving to AIP will increase RIB and FIB entries from 32 bits to 160 bits, with a corresponding increase in the next-hop and each AD component of the path. In addition, for each AD entry (*not* per-peer), the router will require roughly 512 bytes of memory to store its public key.

Table 4 shows the required memory for both IP and AIP, with the cost of that memory as projected from the ITRS roadmap. Unsurprisingly, AIP increases the amount of memory required for RIB storage by about a factor of three. With DRAM cost scaling, however, by 2020 the memory needed to store an AIP table will cost *less* than the memory needed to store a standard IP routing table today.

FIB storage (DRAM, SRAM, or CAM): Router manufacturers appear to use all three common technologies for FIB storage: DRAM, SRAM, and TCAM.[7] As we have seen, the scaling trends for DRAM exceed AIP's growth rate. The results for SRAM are similar: SRAM density is expected to grow 16×, and the FIB will grow only $5-9\times$ (Table 4), leaving adequate room to handle the larger but flat AD entries. The same transistor density scaling affects TCAMs.[8]

Update processing (CPU): The major challenge for update processing is the time required to load a full copy of the table from every peer when a router comes online. This load scales directly with the size of the routing table (Table 1).

Measurements of BGP table load times by Wu et al. show that an older Cisco 3620 router (discontinued in 2003) could process 2,493 updates per second during bulk loads, using a single core 80 Mhz RISC processor [45]. This lower-end router would have required about 48 seconds to load a 120,000 prefix table from each peer, or about *sixteen minutes* for all peers. Scaling that router to a single-core 3.0 GHz modern CPU would suggest a rough 30× speed increase, suggesting that a modern router with a high-performance routing CPU could load 20 peers tables of 240K routing table entries in about 1 minute.

IPv4 routing tables are expected to grow by a factor of between 5 and 9 by 2020 (Table 1). In this period, the number of transistors per CPU is expected to grow by a factor of 16 (Table 3). Disregarding

[7] `http://www.firstpr.com.au/ip/sram-ip-forwarding/router-fib/`

[8] Such large SRAMs could become a "niche" product, resulting in a constant factor increase in their cost relative to other products with the same density. The technology would still be possible, albeit more expensive.

cryptographic overhead for the moment, a future AIP router's CPU should be able to load its full tables in about 30 seconds.

Routing updates may also be constrained by the DRAM bandwidth needed to process them. If an AIP routing table in 2020 requires 8.2 GBytes of memory, even modern DDR2-677 memory (about 1.7 GBytes/s average write bandwidth) could handle the raw load bandwidth requirements in a few seconds.

Cryptographic overhead: Receiving a routing update requires validating an RSA signature. A 2.83 GHz modern quad-core CPU can verify about 35,000 2048-bit RSA signatures/sec and can create 935 signatures/sec, a computation that is both easily parallelized and optimized in hardware.[9] By 2020, a commodity processor should be able to verify 480K and create 13K signatures per second, respectively. This scaling is quite favorable: today, verifying one signature for each route announcement from each of 20 peers would require 164 seconds; by 2020, performing this same verification of entries in an AIP-SBGP table would require about $\frac{1.6M\text{routes}*20\text{peers}}{480{,}000\text{sigs/sec}} = 66$ seconds. Unfortunately, neither of these numbers is quite as good as one might like for table load times, so cryptographic acceleration may still be necessary for either IP with S-BGP or for AIP.

In summary, though some of the modeling in this section is speculative, technology trends suggest that routing scalability with respect to memory consumption, CPU overhead, and network bandwidth are all manageable.

5 Key Management

As with any system that relies heavily on key-based cryptography, AIP faces three general problems in key management:

1. **Key discovery**: Sources must be able to discover the destination address (key).
2. **Individual key compromise**: Domains and hosts must cope with the possibility that their key might be compromised.
3. **Cryptographic algorithm compromise**: In the long term, AIP must be able to migrate to newer digital signature and hash algorithms as earlier algorithms are weakened.

5.1 Key Discovery

Because a host's key is simply its address, the key is obtained automatically once the address is known. Addresses can be obtained as they are today (Section 2.3): manually, using secure or insecure DNS, or using any other lookup service. As a network-layer protocol, AIP is agnostic to the particular lookup mechanism used, though an insecure lookup mechanism presents an obvious avenue of attack.

We also assume that peering `AD`s can identify each other out-of-band. This allows them to exchange public keys in a trusted manner, and also ensure that a compromised key does not lead to misidentified peers (i.e., if an attacker compromises a domain's key, it can't fool another domain into peering with it).

5.2 Key Compromise

There are three issues related to key compromise that we must consider: protecting against compromise, detecting compromise, and dealing with compromise.

The first and third of these are relatively straightforward. To minimize the chance of compromise, hosts and domains should follow established key management practices, such as using time-limited secondary keys for all online signings, and keeping the primary key offline and under strict control. Advances in trusted computing hardware may assist in keeping keys safe.

If a host key is compromised, then the host merely adopts a new key and inserts it into its DNS record in the same manner it inserted its previous key. That is, whatever (possibly out-of-band) authentication and trust mechanisms established its identity with DNS will allow it to change its key.

If a domain key is compromised, then the domain revokes its key through the interdomain routing protocol, and via the public registries discussed below. The only challenge in this scheme is that key revocation must propagate down every path that carries a route for the AD, because after the notice is processed, the route will be withdrawn as invalid.

Beyond these straightforward problems lies a more insidious risk. A very real danger of crypto-based systems such as AIP is one of false confidence: with a compromised key, an attacker could silently impersonate his victim for quite a while before a victim noticed. Much like identity theft in the real world, recovering from a compromise is a hassle, but having it go undetected for a long period can be catastrophic. As a result, we devote the rest of this section to exploring mechanisms to allow both hosts and domains to rapidly detect how and where their identity (key) is being used. While none of these mechanisms other than the domain public key to ID registry are *necessary* for AIP's operation, we believe that some form of these mechanisms would substantially boost the security of the resulting system.

Our answer to this challenge is to maintain a public registry of the peers for each AD and the ADs to which each EID is bound. While the reader may complain that such registries have failed before, the crucial difference here is that these registries only store self-certifying data which has the advantages that:

- There is no need for any central authority to verify the correctness of the registry's content. The registry merely verifies the signature before storing data.
- The registry can be populated mechanistically by the entities involved, with no need for human intervention or involvement. Thus, this approach does not rely on operator vigilance, merely protocol correctness.

We now describe these registries and their use in more detail.

Registries: We assume the existence of global registries where principals can register various cryptographically signed assertions. We also assume the existence of per-domain registries that can be housed by the ISP itself.

Let K_A represent the public key of A. The various classes of assertions, each maintained in a table, are:

- `Keys`: $\{X, K_X\}$

  This table connects a domain or host's address (its public key hash) to its actual public key. There is no need for a signature since an AD or EID X is merely the hash of the corresponding key K_X.

- `Revoked keys`: $\{K_X, \texttt{is\_revoked}\}_{K_X^{-1}}$

  To revoke a key, the key owner inserts an element into this table. Once an entry is written, no further modifications of it are permitted.

[9] Timing from OpenSSL 0.9.8g `openssl speed -multi 4 rsa2048` on an Intel Xeon E5440.

- `Peerings`: $\{A,K_A,B,K_B\}_{K_A^{-1}}$ $\{A,K_A,B,K_B\}_{K_B^{-1}}$

  If A and B are peering, they each sign such a statement and both store it in the registry.

- `ADs of EID` X: $\{A,X\}_{K_A^{-1},K_X^{-1}}$

  When X enters an address $AD:X$ in its DNS record, as discussed below, it must present a certificate from AD that AD is a domain of X. When X asks A for this certificate, A also submits this certificate to the global registry. This table has one entry for each AD that X belongs to.

- First hop router of X: $\{Router,X,MAC_X\}_{K_{Router}^{-1},K_X^{-1}}$

  When a host registers with its first-hop router, the first-hop router registers this fact in a domain-wide database (per-domain registry). It also registers X's MAC address.

Clients can do a lookup using the hash of the key to find the relevant data.

Maintaining the domains registry: One challenge is keeping the domains registry reasonably up to date. A compromised host has no incentive to list itself, so this responsibilities lies with the `AD` providing a home for the compromised `EID`. To encourage such behavior, we propose forcing the domain to sign $A:X$ entries before the DNS servers and resolvers will accept them as the result of a DNS resolution. Any client who performs a DNS lookup, then, can with confidence insert the results in the domains registry. A DNS entry thus must bind name $\rightarrow \{AD:EID\}_{K_{AD}^{-1},K_{EID}^{-1}}$. A hierarchical address must be signed by all domains in the address.

Using the registries: These registries are used by both domains and hosts to check for compromise.

- Each host X periodically checks a global registry for which domains are hosting it, and checks its domain-specific registry for which first-hop routers are hosting it. If it sees an entry it doesn't recognize, it may assume there has been a compromise.
- Each domain A periodically checks the global registry to see which domains claim to be peering with it. If it sees an entry it doesn't recognize, it may assume there has been a compromise.

Using these mechanisms, a domain can recognize whenever an imposter has established a peering arrangement with some other domain. Because we are assuming that out-of-band mechanism can prevent an imposter from fooling a peer (that is, if A thinks it is peering with B, it can verify the identify of the peering entity in ways other than verifying B's signature), then there is no way an imposter can enter the interdomain routing system without the valid domain being able to detect its presence.

Similarly, a host can recognize whenever an imposter has established itself in another domain, or in the same domain with another first-hop router, or at the same first-hop router with a different MAC address. The only case these mechanisms don't cover is when an imposter registers with the same first-hop router with the same MAC address. Dealing with this latter case would require L2 security technologies, which are outside of our scope.

5.3 Cryptographic Algorithm Compromise

To cope with the inevitable compromise of existing cryptosystems and hash functions, each AIP address (src, dst, every AD in the stack, EID, and so on) and every registry entry (as described above) contains its own crypto version field (Figure 1). Versioning each address separately is necessary to support gradually phasing in new algorithms. Because of the large number of stakeholders that must agree on a shared set of signature algorithms and hash functions, a particular crypto version represents *one* combination of a signature scheme and hash function. For example, in our design, crypto version 0 represents RSA signatures with SHA-1 hashes. The hash function must be truncated or zero-filled, as appropriate, to fill the 144-bit hash space in the AIP address.

We envision that at most two or three crypto versions will be present on the network at any given time: the "legacy" version that the network is moving away from, and the newer algorithm that is supplanting it over five or ten years.

6 Traffic Engineering and AD Size

The goal of traffic engineering is to map an offered load on to a set of available paths. This operation happens in two ways in today's Internet: per-prefix, and per-service. Network operators remap load by selectively advertising prefixes to control how traffic destined for groups of hosts flows on various paths. Server administrators use DNS mappings to direct traffic to individual hosts. By moving away from prefixes, AIP forces a reconsideration of these issues.

A fundamental difference between AIP and IP is that `AD`s cannot be split into sub-prefixes for finer control over routing. Because of this limitation, we must answer three questions to assess how to perform common traffic engineering functions and how debilitating the elimination of prefixes might be.

1. What is the granularity of an AD?
2. Will operators want to "split" an AD in order to better perform traffic engineering? How can AIP support this?
3. How does DNS-based load balancing work under AIP?

AD granularity: As an accountability domain, we envision an AD as corresponding to a group of nodes that meets two criteria: they are administered together, and they would fail together under common network failures. For example, ADs might represent a campus, a PoP, or a single non-geographically-distributed organization. This assignment also helps reduce false churn, as this granularity corresponds roughly to the way in which connectivity to the network changes.

Splitting ADs for TE: The PoP and customer site granularity of ADs is a good match with ISPs' typical traffic engineering goals, where operators often wish to control traffic flow at the granularity of PoP-to-PoP traffic across their core network (e.g., using MPLS). Because ADs are assigned at the granularity of a single campus or sub-network reachable via one connection, they are a good match for existing inbound traffic engineering techniques.

To a first approximation, creating an AD from each prefix in the wide-area BGP routing tables seems like a reasonable strategy. This approach, however, prevents a network operator from unilaterally advertising sub-prefixes of a prefix P to different upstream routers all belonging to the same ISP, relying on that ISP to aggregate the sub-prefixes to prevent them from reaching the "global" routing tables. With AIP as presented thus far, accomplishing such traffic engineering does not seem possible without increasing the number of globally visible AD entries.

We believe, however, that splitting and then aggregating prefixes for traffic engineering is not widely practiced today. We conducted a measurement study by obtaining a /17 and advertising various sub-prefixes of size /18 and /19 via BGP. Each sub-prefix had the

same AS path and IP layer path, so an upstream AS could easily aggregate these sub-prefix advertisements into a single one. We found by examining BGP routing tables at many Internet vantage points that no aggregation occurred, despite many different ISPs having had the opportunity to do so. Private conversations with some ISP operators confirmed our finding: such explicit aggregation is almost never done today.

To explain why, we conducted a related experiment. We announced a /18 via two large Tier-1 ISPs, A and B, and the containing /17 from just one of them (B) to emulate the case where an upstream (the /17) owns a larger chunk and a stub is punching a hole to multihome. The /18 announced to the two upstream ISPs was seen by all of the many BGP vantage points we checked. Using traceroute, we found that some nodes in the Internet reached the /18 via A and some via B. If A had aggregated the /18s into a single /17, then longest-prefix matching would likely cause all of the traffic to arrive via B. Such unilateral filtering and aggregation could be harmful and violate the stub network's traffic engineering goals.

We then withdrew the /18 through B, simulating the case when the link between the stub and its primary fails. If some AS had filtered the /18s earlier, it would not have seen the withdrawal of the /18 through B, and hence continued to use that route to get to destinations in the withdrawn /18. We found instead that all subsequent traceroutes took the valid route through A. This result suggests that ISPs today do not arbitrarily filter route advertisements even when they might be redundant; one reason might be to avoid failing in situations like the one just described. Thus, it appears that a stub network's BGP announcements remain invariant as they propagate across the Internet. The simple approach of making each of today's prefixes an AD might well be adequate for interdomain traffic engineering.

If, however, operators wish to use such load balancing in the future, one can make use of the AIP address interface bits to subdivide an `AD`, so an operator could announce different routes for each AD:interface bits combination. In particular, an administrator could partition the EIDs belonging to the different subnetworks of an AD to one of 255 possible "paths" (in practice, we expect the number of paths to be a lot smaller) by setting a different value of the interface field for each partition. Any upstream network can choose to zero the interface bits in its wide-area advertisements as a way to reduce the amount of global routing table state, in much the same way that it aggregates prefixes today. Thus, most wide-area routers will forward using AD lookups as before, but those near the destination that might be obligated to pay attention to the interface bits of an AD address. It is important to note here that BGP messages are still signed using the AD's private key, and that there remains only one key pair for the entire AD. Finally, we note that AIP leaves unchanged an operator's ability to perform equal-cost multipath routing by hashing packet header fields to direct traffic to one of k links.

DNS-based load balancing: Another component of traffic engineering is a service-centric view: How to load balance traffic destined for a particular service across machines in a cluster or across data centers. In general, the move to AIP has only a small, positive effect on this ability. DNS-based load balancing can still change the AIP-record corresponding to a host or service name. AIP's interface bits might simplify cluster-based load-balancing by representing a service as a single "host" connected to the network multiple times; we leave the design of such specific mechanisms to future work.

7 Related Work

Many features of AIP have similar forerunners in the literature. We do not have space to discuss each predecessor individually, but below we list some of the major ideas that AIP draws upon.

Self-certifying names: Some forms of addressing are already self-certifying: CGA [5] derives the "interface" portion of an IP address from a public key, and HIP [29] derives a host identifier (EID) from the hash of a public key. AIP extends self-certification to the entire network-level address.

Separating identifiers and locators: Ever since the GSE/8+8 [30] proposal, it has been widely acknowledged that addressing should separate identification from location. The more recent LISP proposal also suggests forwarding traffic based on routing locators that are separate from endpoint identifiers [9]. AIP provides a similar separation of function: The AD provides location information, while the EID is purely an identifier. Unlike LISP, AIP does not require tunneling to route on locators, and AIP's addresses are self-certifying.

Scalability: The scalability of Internet routing has been a long-standing concern. CIDR has helped sustain the routing table and control traffic load growth until increases in peering, but site-multihoming, and preference for provider independent addresses have combined to reduce CIDR's effectiveness. A number of recent proposals have tried to re-establish aggregation by introducing address aggregates in the control plane to reduce its load (e.g., [17]) or in both the control and forwarding planes [41, 46] to limit the growth of routing tables. The challenge of finding a mechanism for aggregation in Internet has sparked theoretical interest as well [21].

Source accountability: Several mechanisms have been proposed to improve source accountability. The simplest mechanism is to install filters on the border routers of an AS [10, 20]: routers can prohibit outbound traffic originating from source addresses that do not exist on the local network (egress filtering) and can prohibit inbound traffic from source addresses that should only exist inside the local network (ingress filtering). To be effective, filtering requires near-complete deployment by stub networks. Installing these filters in the middle of the network is fundamentally difficult [8, 31]. Passport [25] uses cryptography to prevent spoofing; it verifies each packet but only at the domain granularity. Whereas AIP is based on a challenge-response for an initial packet and operates on a per-host granularity,

As mentioned earlier, AIP's shut-off packet is based on a similar idea proposed by Shaw [35]. It also bears some similarity to AITF [4]. Thus, we don't claim credit for the basic idea, but merely observe that AIP's address structure facilitates its implementation.

Control-plane accountability: Various proposals augment the routing protocol with cryptographic mechanisms to prevent routers from originating false routes; the two most notable are S-BGP [19] and secure origin BGP (soBGP) [44]. One of the major drawbacks to these proposals is that they require the existence of a global PKI. Although work is afoot to institute such a framework [15, 14, 3], it may be difficult to maintain in practice, as has been the case with routing registries [13, 36]. Other mechanisms for securing BGP include online monitoring at global BGP routing repositories to detect suspicious routing announcements [16, 22, 24, 34]. Although detecting "suspicious" routing announcements in real-time is operationally useful and is attractive due to its low deployment barrier, these approaches are more effective for detecting accidental misconfiguration than for preventing malicious attacks.

8 Conclusion

AIP is our attempt to answer the question: "what might the Internet's network layer look like if accountability were a first-order goal?" By using a simple hierarchical addressing scheme with self-certifying components, AIP enables simple solutions to source spoofing, shutting off certain kinds of DoS traffic, and securing BGP. The move away from prefixes to flat addresses brings up concerns about route scalability and traffic engineering, while the use of self-certification raises questions of key management and compromise. Our discussion of these issues leads us to conclude that, while hoping for a near-term replacement to IP might be like building castles in the air, these important concerns are not a show-stopper for AIP (or the ideas contained therein) to be widely adopted.

Acknowledgments

We thank Mythili Vutukuru for conducting the traffic engineering experiments reported in Section 6. We thank Mythili and Michael Walfish for participating in the early stages of the AIP project and contributing to its design. Jakob Eriksson, Lewis Girod, Ramakrishna Gummadi, Adrian Perrig, Amar Phanishayee, Anirudh Ramachandran, Vijay Vasudevan, the attendees of the 6th HotNets workshop, and the anonymous reviewers provided helpful comments that improved the paper. This work was funded in part by the National Science Foundation under awards CNS-0716273 and CNS-0520241.

References

[1] ITRS international technology roadmap for semiconductors, 2006.

[2] D. Andersen, H. Balakrishnan, N. Feamster, T. Koponen, D. Moon, and S. Shenker. Holding the Internet accountable. In *Proc. 6th ACM Workshop on Hot Topics in Networks (Hotnets-VI)*, Nov. 2007.

[3] APNIC. The APNIC Resource Certification Page. `http://mirin.apnic.net/resourcecerts/`.

[4] K. Argyraki and D. R. Cheriton. Active Internet traffic filtering: Real-time response to denial-of-service attacks. In *Proc. USENIX Annual Technical Conference*, Apr. 2005.

[5] T. Aura. *Cryptographically Generated Addresses (CGA)*. Internet Engineering Task Force, Mar. 2005. RFC 3972.

[6] R. Beverly and S. Bauer. The Spoofer project: Inferring the extent of source address filtering on the Internet. In *Proc. SRUTI Workshop*, July 2005.

[7] CNET News.com. Router Glitch Cuts Net Access. `http://news.com.com/2100-1033-279235.html`, Apr. 1997.

[8] Z. Duan, X. Yuan, and J. Chandrashekar. Constructing Inter-Domain Packet Filters to Control IP Spoofing Based on BGP Updates. In *Proc. IEEE INFOCOM*, Mar. 2006.

[9] D. Farinacci, V. Fuller, D. Oran, and D. Meyer. *Locator/ID Separation Protocol (LISP)*. Internet Engineering Task Force, Apr. 2008. Internet Draft (`http://tools.ietf.org/html/draft-farinacci-lisp-07`). Work in progress, expires October 2008.

[10] P. Ferguson and D. Senie. *Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing*. Internet Engineering Task Force, Jan. 1998. RFC 2267.

[11] P. Ferguson and D. Senie. *Network Ingress Filtering*. Internet Engineering Task Force, May 2000. BCP 38, RFC 2827.

[12] V. Fuller. Scaling issues with routing+multihoming, Feb. 2007. Plenary session at APRICOT, the Asia Pacific Regional Internet Conference on Operational Technologies.

[13] G. Goodell, W. Aiello, T. Griffin, J. Ioannidis, P. McDaniel, and A. Rubin. Working around BGP: An incremental approach to improving security and accuracy in interdomain routing. In *Proc. NDSS*, Feb. 2003.

[14] G. Huston, G. Michaelson, and R. Loomans. *A Profile for Resource Certificate Repository Structure*. Internet Engineering Task Force, June 2006. `http://mirin.apnic.net/resourcecerts/project-notes/draft-ietf-sidr-repos-struct-00.html`.

[15] G. Huston, G. Michaelson, and R. Loomans. *A Profile for X.509 PKIX Resource Certificates*. Internet Engineering Task Force, July 2006. `http://mirin.apnic.net/resourcecerts/project-notes/draft-ietf-sidr-res-certs-02.html`.

[16] J. Karlin, S. Forrest, and J. Rexford. Pretty Good BGP: Protecting BGP by cautiously selecting routes. Technical report, University of New Mexico, Oct. 2005. TR-CS-2005-37.

[17] F. Kastenholz. *ISLAY: A New Routing and Addressing Architecture*. Internet Engineering Task Force, May 2002. `http://ietfreport.isoc.org/idref/draft-irtf-routing-islay/`.

[18] S. Kent and R. Atkinson. *Security Architecture for the Internet Protocol*. Internet Engineering Task Force, Nov. 1998. RFC 2401.

[19] S. Kent, C. Lynn, and K. Seo. Secure border gateway protocol (S-BGP). *IEEE JSAC*, 18(4):582–592, Apr. 2000.

[20] T. Killalea. *Internet Service Provider Security Services and Procedures*. Internet Engineering Task Force, Nov. 2000. RFC 3013.

[21] D. Krioukov, kc claffy, K. Fall, and A. Brady. On Compact Routing for the Internet. *ACM Computer Communications Review*, 37(3):41–52, July 2007.

[22] M. Lad, D. Massey, D. Pei, Y. Wu, B. Zhang, and L. Zhang. PHAS: A prefix hijack alert system. In *Proc. 15th USENIX Security Symposium*, Aug. 2006.

[23] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: Densification laws, shrinking diameters and possible explanations. In *Proc. 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug. 2005.

[24] J. Li, R. Bush, Z. M. Mao, T. Griffin, M. Roughan, D. Stutzbach, and E. Purpus. Watching data streams toward a multi-homed sink under routing changes introduced by a BGP beacon. In *Passive & Active Measurement (PAM)*, Mar. 2006.

[25] X. Liu, X. Yang, D. Wetherall, and A. Li. Passport: Secure and Adoptable Source Authentication. In *Proc. 5th USENIX NSDI*, Apr. 2008.

[26] D. Mazières, M. Kaminsky, M. F. Kaashoek, and E. Witchel. Separating key management from file system security. In *Proc. 17th ACM Symposium on Operating Systems Principles (SOSP)*, pages 124–139, Dec. 1999.

[27] D. McCullagh. How Pakistan knocked YouTube offline. `http://news.cnet.com/8301-10784_3-9878655-7.html`, Feb. 2008.

[28] D. Meyer, L. Zhang, and K. Fall. *Report from the IAB Workshop on Routing and Addressing*. Internet Engineering Task Force, Sept. 2007. RFC 4984.

[29] R. Moskowitz and P. Nikander. *Host Identity Protocol (HIP) Architecture*. Internet Engineering Task Force, May 2006. RFC 4423.

[30] M. Ohta. *8+8 Addressing for IPv6 End to End Multihoming*, Jan. 2004. draft-ohta-multi6-8plus8-00 (Expired IETF Draft).

[31] K. Park and H. Lee. On the effectiveness of route-based packet filtering for distributed DoS attack prevention in power-law Internets. In *Proc. ACM SIGCOMM*, Aug. 2001.

[32] A. Ramachandran and N. Feamster. Understanding the Network-Level Behavior of Spammers. In *Proc. ACM SIGCOMM*, Aug. 2006. An earlier version appeared as Georgia Tech TR GT-CSS-2006-001.

[33] A. Ramachandran and N. Feamster. Understanding the network-level behavior of spammers. In *Proc. ACM SIGCOMM*, Aug. 2006.

[34] Renesys. Renesys Routing Intelligence. `http://www.renesys.com/products_services/routing_intelligence.shtml`.

[35] M. Shaw. Leveraging good intentions to reduce unwanted network traffic. In *Proc. USENIX Steps to Reduce Unwanted Traffic on the Internet workshop*, July 2006.

[36] G. Siganos and M. Faloutsos. Analyzing BGP Policies: Methodology and Tool. In *Proc. IEEE INFOCOM*, Mar. 2004.

[37] T. L. Simon. oof. panix sidelined by incompetence... again. `http://merit.edu/mail.archives/nanog/2006-01/msg00483.html`, Jan. 2006.

[38] A. C. Snoeren and H. Balakrishnan. An end-to-end approach to host mobility. In *Proc. ACM Mobicom*, pages 155–166, Aug. 2000.

[39] Spammer-X. *Inside the SPAM Cartel*. Syngress, 2004. Page 40.

[40] G. Varghese. *Network Algorithmics*. Morgan Kaufmann, 2007.

[41] P. Verkaik, A. Broido, kc claffy, R. Gao, Y. Hyun, and R. van der Pol. Beyond CIDR aggregation. Technical Report TR-2004-01, CAIDA, Feb. 2004.

[42] Q. Vohra and E. Chen. *BGP Support for Four-octet AS Number Space*. Internet Engineering Task Force, May 2007. RFC 4893.

[43] M. Walfish, J. Stribling, M. Krohn, H. Balakrishnan, R. Morris, and S. Shenker. Middleboxes no longer considered harmful. In *Proc. 6th USENIX OSDI*, Dec. 2004.

[44] R. White. Securing BGP through secure origin BGP. *The Internet Protocol Journal*, 6(3), Sept. 2003. `http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_6-3/ipj_6-3.pdf`.

[45] Q. Wu, Y. Liao, T. Wolf, and L. Gao. Benchmarking BGP routers. In *Proc. IEEE International Symposium on Workload Characterization (IISWC)*, Sept. 2007.

[46] X. Zhang, P. Francis, J. Wang, and K. Yoshida. Scaling IP routing with the core router-integrated overlay. In *IEEE International Conference on Network Protocols (ICNP)*, Nov. 2006.