Holding the Internet Accountable

David Andersen, Hari Balakrishnan, Nick Feamster,
Teemu Koponen, Daekyeong Moon, Scott Shenker

Abstract

Today's IP network layer provides little to no protection against misconfiguration or malice. Despite some progress in improving the robustness and security of the IP layer, misconfigurations and attacks still occur frequently. We show how a network layer that provides *accountability*, *i.e.*, the ability to associate each action with the responsible entity, provides a firm foundation for defenses against misconfiguration and malice. We present the design of a network layer that incorporates accountability called **AIP** (Accountable Internet Protocol) and show how its features—notably, its use of self-certifying addresses—can improve both source accountability (the ability to trace actions to a particular end host and stop that host from misbehaving) and control-plane accountability (the ability to pinpoint and prevent attacks on routing).

1 Introduction

The Internet architecture is elegant and simple; it has been a tremendous success. When judged from the perspective of security, however, the architecture has several serious shortcomings. The IP layer, in particular, is riddled with vulnerabilities. Denial-of-service (DoS) attacks occur daily and are launched with impunity because attackers are confident they will neither be identified nor thwarted easily. Misconfigurations can cause massive Internet outages (as in the infamous AS 7007 incident [4] and similar occurrences in recent years [26]). Route hijacking is used to send untraceable spam [23].

Our goal is to develop a network architecture to address these IP layer vulnerabilities. Our work does not enter a vacuum, as there is a profusion of network security solutions addressing the set of problems discussed in this paper. However, these solutions tend to be narrowly focused, and their union does not form a coherent architecture. We can be fairly certain that if these solutions were implemented and deployed, IP would be more secure. Unfortunately, it is also clear that IP would have lost much of its simplicity, elegance, and coherence.

To achieve both practical security and architectural coherence, we need to develop a simple foundation upon which security solutions can be easily built. In searching for a unifying principle, we turn to daily experience. In the real world, security relies on *accountability*.[1] In a just and secure society, one ought to be able to prove that a guilty party did in fact commit a crime. Equally importantly, framing an innocent party should be difficult. Having such a framework for accountability, together with suitable laws and punishment, is important to provide adequate incentives for crime prevention.

We believe the same holds true for computer systems. In order to identify, isolate, prevent, and possibly punish bad behavior (*e.g.*, by not providing service to wrong-doers), a computer system must have some notion of accountability. Indeed, one of the reasons it has been so hard to secure the Internet is that the IP architecture provides very little accountability. For instance, IP addresses do not always map one-to-one to Internet end points because of NATs, firewalls, and proxies. More dangerously, IP addresses are trivial to forge, so attempting to use IP addresses to identify or ignore wrong-doers opens up another attack vector.

This paper presents **AIP** (Accountable Internet Protocol), a network architecture that provides accountability as a first-order principle and serves as a building block for simple, elegant improvements to Internet security. AIP's cornerstone is *self-certifying addressing*. AIP addresses are of the form `AD:EID`, where `AD` is the identifier for the autonomous domain that the host belongs to, and `EID` is a globally unique host identifier. Both address components are derived from public keys held by the domain and host, respectively. The structure of these addresses allows other entities to verify the authenticity of routing messages and the provenance of data packets.

AIP addresses enable protocols to (i) detect and prevent spoofing or forgery of source addresses (*source accountability*); (ii) throttle certain forms of unwanted traffic using a simple "shut-off message", taking advantage of the source accountability property to ensure authenticity; and (iii) detect misleading route advertisements (*control-plane accountability*). AIP aims to add accountability with only small changes (and minimum additional complexity) to the network architecture; it therefore can, and should, co-exist

1. We focus on accountability in the sense that we wish to attribute actions to the responsible party; we do *not* mean accounting, and have no intention of providing facilities for resource allocation.

with mechanisms that provide other important features (mobility, higher availability, etc.).

In the next section, we describe these two forms of accountability. We then give an overview of the AIP architecture in Section 3. We discuss control plane accountability and source accountability in Sections 4 and 5. In Section 6 we discuss three design challenges raised by AIP: scaling, traffic engineering, and key compromise.

2 Why Accountability?

AIP makes accountability a first-order design goal. By doing so, it improves the security, trustworthiness, and robustness of both the IP layer and of the systems built atop it. Accountability makes certain types of attacks either more traceable or simply more difficult to mount. AIP provides two types of accountability: control-plane accountability and source accountability. For each, we briefly discuss how the absence of accountability leads to insecurity and how its presence might provide a foundation upon which robust security measures could be built.[2]

Control-plane accountability: If routers and ASes were accountable for their routing messages, then their peers would be able to more easily discover forgeries or errors. Ultimately, securing a protocol like BGP (as in S-BGP [14]) relies on ensuring that no entity can undetectably forge routing messages, which is actually a statement about accountability.

Securing routing is difficult in part because today's IP addressing structure does not securely bind addresses to the networks that are allowed to announce them. S-BGP provides mechanisms to do so, but we believe that such a binding should be *inherent* to the network architecture.

Source accountability: Today's Internet architecture lacks source accountability: hosts can easily forge the source IP address of data traffic, which makes attacks difficult to track and makes it nearly impossible for network operators to filter traffic based on the source address—the most logical identifier for doing so.

If sources were accountable, then any element in the network that saw a packet could verify that packet's origin. This property eliminates undetectable source address forgery. As a result of preventing such forgery, defenses against DoS could profile using source addresses, routers could implement packet filters or rate limiters using source addresses as a robust handle, spam filters could more easily blacklist on IP addresses, and intrusion detection and prevention systems could use source addresses as a handle to their state without worrying about adversaries forging source addresses and exhausting their state.[3]

In this paper, we do not address data-plane accountability (*i.e.*, ways to identify network elements that are not forwarding packets appropriately). Several aspects of AIP, however, do facilitate solutions to this problem. First, all elements in the architecture have strong, verifiable identities that can serve as a basis for attestations about behavior. Second, hosts' globally unique identifiers (Section 3.2) facilitate avoiding data-plane failures through multihoming or by continuing connections through different ADs or interfaces. Finally, AIP's self-certifying address structure makes it possible to give cryptographic assurance to mechanisms that attribute packet loss, delay, or misrouting to a network element; one logical candidate mechanism is AS-level packet obituaries [1].

3 Accountable Internet Protocol

This section outlines AIP, starting with the structure and function of AIP addresses. We then explain how making AIP addresses self-certifying (*i.e.*, derived from public keys) infuses accountability into the network layer.

3.1 Basic Structure and Function

The Internet's original addressing structure was a simple two-level hierarchy. Each address had a network and a host component, and routers inspected only the network portion until the packet reached the destination network. The network and host components were both implicitly assumed to be flat: there was little correlation between topological and numerical proximity. Unfortunately, addressing has become more complicated with the advent of autonomous systems (used in BGP routing) and classless routing (CIDR); these changes have made it hard to add accountability to the existing infrastructure.

Address structure: AIP returns to simple two-level hierarchical addresses. We assume that there are some number of independently administered networks (as is the case today) which we call *autonomous domains* (ADs), and that each possesses a unique identifier. We avoid the term "AS" because we envision that current large ASes would be broken up into several smaller ADs for traffic engineering (Section 6.2). Each host is assigned a unique endpoint identifier (EID). Analogous to the original Internet addressing structure, the AIP address of a host currently homed in

2. Accountability, as we have defined it, does not preclude anonymity for end-to-end applications (such as anonymous email), which can be provided using approaches such as onion routing and mix-nets.

3. The simplicity of profiling and filtering is in stark contrast to the more radical architectural designs needed to deal with DoS in the presence of spoofing, such as [15, 11, 27, 32].

some AD would have an address of the form AD:EID.[4] The EID is a globally unique endpoint identifier, and it is part of the internetwork address (as in IPv6 CGA [2]).

Name lookup: The domain name system would include an AIP-record, which would contain 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. In addition, to allow even more fine-grained control of traffic for the host, EIDs could be augmented with *interface bits* that give each interface a unique identifier: EIDif_1, EIDif_2, etc. Each of these identifiers would appear in the host's AIP-record.

Interdomain routing: In AIP, interdomain routing occurs in much the same way that it does today (and can benefit from improvements to BGP). Rather than involving IP prefixes, however, *interdomain routing occurs entirely at the AD granularity*, so the only advertisements will be for ADs themselves. Interior and border routers in an AD maintain routing information on a per-AD basis for destinations in other ADs; *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 [20]) to maintain routing information to the EIDs within the AD. Although AIP changes the granularity of routing, it does not specify or mandate any particular inter- or intra-domain routing protocol.

Packet forwarding: Packets contain the destination's AD:EID. 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.

3.2 Self-Certifying Addresses

Eliminating structure in the AD and EID allows us to make them *self-certifying*. 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) [18, 30], and HIP [19] 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.

Security should not rely on extensive manual configuration or globally trusted authorities. Thus, we believe that self-certification is an indispensable aspect of providing accountability at the network layer. Accountability requires a verifiable identity, and in these settings the only practical method of verification uses cryptographic signatures. Thus, identifiers must be bound to their public key.

Existing schemes (*e.g.*, S-BGP [14]) implement this binding using registries that map identifiers to their public keys (a PKI). Unfortunately, these registries must be both up-to-date (via manual configuration[5]) and globally trusted. Self-certifying addressing frees security mechanisms from undesirable trust relationships or manual configuration. Existing IP and transport security mechanisms (*e.g.*, IPsec [13]) could also use AIP's self-certifying address structure to securely establish the identity of a remote host, rather than relying on an external infrastructure.

4 Control-Plane Accountability

Today's Internet routing infrastructure provides almost no accountability. A malicious or misconfigured AS can "hijack" IP address space (*i.e.*, advertise IP prefixes that it does not own), because there is no intrinsic association between an autonomous system number and its part of the IP address space. It is also possible to forge routing announcements with false AS paths, causing traffic to be redirected in unexpected and undesirable ways.

Securing routing involves fixing both of these problems: the infrastructure should provide *origin authentication* (ensuring that the AS that originated the route actually owns the block of IP addresses being advertised) and *path authentication* (ensuring the accuracy of the AS path). Solving these problems involves holding an AS accountable for its routing announcements.

Recent proposals retrofit cryptographic mechanisms onto BGP to address these shortcomings [14, 31]. Unfortunately, the deployment of such secure routing schemes is hindered by the Internet's simultaneous use of two logically distinct and unrelated name spaces: AS numbers and IP prefixes. This independence forces routing security to depend on *two external infrastructures:* a "routing registry" (a database recording which AS owns each prefix) and a PKI for ASes. Even if a PKI for ASes' public keys came to pass, experience has shown that such types of registries are disappointingly inaccurate [10].

Self-certifying ADs make secure BGP routing intrinsic to the architecture, not dependent on external registries or operator vigilance. Origin and path authentication are natural by-products of this design, as we now show.

Origin authentication, which is particularly lacking in the current routing system, becomes automatic because

4. We expect that each field would be at least 128 bits in size.

5. Experience with Internet address registries suggests that keeping these registries accurate and up-to-date will be difficult [10, 12, 25].

AD numbers are derived from public keys. ADs can exchange public keys using separate BGP messages or using a lookup service [31]. Verifying AD announcements appears practical, given current cryptographic speeds: a router that learns several hundred thousand ADs and two or three routes per AD could verify an entire routing table's worth of signatures on the order of minutes with manageable computational overhead.[6]

Path authentication proceeds as in S-BGP [14]: some router in AD_i signs the AD path $[AD_{i+1}\ AD_i\ \ldots\ AD_0]$. A router receiving an announcement with this AD path verifies every signature in the route update before installing the route in its routing table. Thus, each route advertisement must be signed once by each AD along the AD path; a router that receives a route must verify $N-1$ signatures, where N is the number of labels on the AD path. The significant advantage over the status quo is that because AD identifiers are self-certifying, path authentication does not need a PKI. One disadvantage to this approach is that revoking a public key requires the AD number to change. Section 6 considers this shortcoming in more detail.

5 Source Accountability

AIP provides an automatic mechanism for source accountability, ensuring that hosts cannot spoof the source address of their packets. This mechanism enhances the effectiveness of some current schemes to combat DoS attacks (*e.g.*, by filtering [11] or simply by contacting the ISP responsible for the offending traffic) and also enables new defenses, as we discuss later in this section. The inability to trust the source address of packets has caused numerous other security vulnerabilities, such as those due to using trusted IP addresses for authentication in ".rhosts" files. In general, we believe that *many other aspects of the network architecture pertaining to security will be improved or simplified if they need not deal with address spoofing*.

5.1 Preventing Spoofing

The limited success of ingress filtering [3, 7] has shown that mechanisms that depend on correct operator action are often only marginally effective. AIP's source accountability, in contrast, makes use of self-certifying addressing to develop simple mechanisms that verify the source of packets—and drop the packets if the sources are spoofed. AIP's source accountability mechanism requires no configuration by operators and no interaction from operators or end-users.

AIP's source accountability mechanism extends (and makes feasible) "unicast reverse path forwarding" (uRPF) [8]. uRPF is an automatic filtering mechanism that only accepts packets if the route to the packet's source

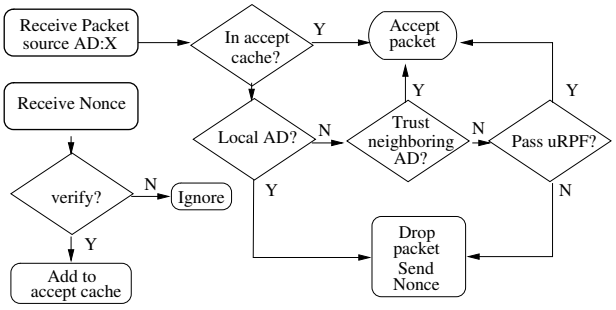


Figure 1: Process for verifying a packet's source address.

address points to the same interface on which the packet arrived. uRPF is extremely useful at the edge for automatically preventing 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 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 1, is as follows:

EID verification: If the first-hop router, R, has not recently verified the source host, it drops the packet and sends a nonce to the source. To prevent the router R from needing to keep state for each nonce it sends, this nonce is a function of the source's `EID`, a coarse timestamp, and a secret known only to R. The sender must prove that it has identity `EID` by signing the nonce with the private key associated with `EID`. If the host produces the correct signature, R caches this information and allows subsequent packets to pass. The host must re-send ("reflect") the packet that generated the nonce to the router, to avoid having any router maintain state for 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. In fact, this process

6. Cryptographic hardware could, of course, reduce this time considerably, but even with only commodity processors the time is not excessive. Route processors are getting faster (Cisco's CRS-1 route processor is 1.2GHz); signature verification can be performed offline; and fast cryptosystems such as ESIGN [22] running on a 3 GHz processor can create and verify 2048-bit signatures in 150 and 100 microseconds, respectively [17, §7.2.2].

is analogous to a "verifiable ARP" check, and can also be used to prevent sources from forging ARP replies.

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 set of 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, which determine whether it arrived on an interface leading back to that source. If the packet's source address passes uRPF, *A* forwards the packet.
3. If these tests fail (*e.g.*, in the case of route asymmetry), *A* drops the packet and sends a nonce to `AD:EID`, asking `EID` to produce a signature. If EID does so, it proves two things: First, that `EID` originated the packet that triggered the nonce exchange. Second, that `EID` is legitimately contained in `AD` (or in one of `AD`'s upstream networks, which is in a position to spoof packets on behalf of `EID` in *any* case) and so should be allowed to transmit packets as `AD:EID`.

To avoid keeping state, *A* generates the nonce in a manner similar to a "SYN cookie", hashing a coarse time counter t, the source address `AD:EID`, and a secret s known only to *A*. *A* can then verify that the returned nonce is correct, using the timestamp to prevent replay attacks. If it is, then *A* temporarily inserts `AD:EID` into its accept cache, allowing subsequent packets through.[7]

Preventing address "minting": While the mechanisms above prevent an attacker from spoofing its AD or using the EID of another host, an attacker could mint new addresses at will. This minting could be used to circumvent EID-based filtering. By preventing AD-level spoofing, however, AIP creates an incentive for ADs themselves to prevent or limit minting. A victim facing an attack from an inordinately large number of (apparently) unique EIDs from one domain would simply filter all traffic from that domain, stopping the attack but potentially causing collateral damage. ADs can easily limit the number of EIDs a host can claim using techniques similar to MAC-address limiting in Ethernet, or prevent EID forgery by having internal mechanisms to authorize EIDs to use particular ports.

5.2 "Shutting-Off" Unwanted Traffic

While the techniques above directly eliminate some classes of DoS attacks, such as "reflector" attacks that forge requests to appear to originate from the victim, other attacks remain unaffected, such as flooding a victim with traffic from compromised hosts. AIP's self-verifying addresses enable a new approach to throttling unwanted traffic whereby a victim host sends an explicit "shut-off" message to a host sending it traffic that it doesn't want to receive. Self-verification ensures the authenticity of these messages.

Most compromised machines are owned by well-intentioned users or businesses [24]. Although the vulnerabilities caused by the complexity of modern software make it difficult for the owners to prevent compromises, they do not launch attacks of their own volition. We therefore envision equipping end hosts with a smart network interface card ("smart-NIC") that aids in controlling the network behavior of the end host by selectively suppressing or rate-limiting packet transmission. The suppression mechanism of the smart-NIC would be beyond the reach of the host OS and thus wouldn't be subject to compromise. The only way to modify the NIC's firmware or configuration would be by having physical access to it, *e.g.*, by plugging it into a USB or serial interface. In normal operation, it would be unmodifiable from the host.

The smart-NIC would record the hashes of recently sent packets and respond to a special class of packets called *shut-off packets* (SOPs). A SOP sent from host X to host Y would include X's public key, a hash of a recent packet sent to X from Y, and a TTL, all signed by X. Upon receiving such a packet, the smart-NIC would first check to see if it indeed had any record of such a packet. If not, it would disregard the SOP; if so it would install a filter in the NIC suppressing further packets from Y to X for the duration of the TTL.

AIP's combination of self-certifying addresses and spoof detection makes this approach feasible. X's signature and its key (which can be verified as belonging to X) assures 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, and to prevent an attacker from exhausting the filter state in the NIC to allow them to continue attacking a chosen victim. It is important that this process not require a three-way handshake, as a host under attack may not receive the return packets. This approach can also be extended to filtering packets sent to an AD, not just an individual host.

7. We depict the accept cache as examining both AD and EID, so that one rogue host cannot give outsiders the ability to spoof other nodes in its AD. For efficiency, of course, a router might choose to weaken this property by keying the accept cache on a per-AD basis.

6 Challenges

AIP's structure is conceptually simple, but significant research will be needed to ensure that it will work in practice. We outline three specific questions about AIP with regard to practical issues of scaling, traffic engineering, and key management.

6.1 Routing Scalability

Only 15 years after the Internet moved away from classful routing, why do we suspect that flat addressing can scale as well *or better* than current Internet routing? We examine this question along two axes: the amount of routing *state* and the *volume* of routing updates.

The routing state that AIP must cope should be manageable. Modern routers handle one million or more routing table entries [29], and with sufficient incentive, vendors could scale this number without too much difficulty. Furthermore, routing on ADs requires a flat table look-up, not the more complex longest-prefix match required under today's IP architecture. AIP might increase routing table state, because we do propose splitting some large ASes into several ADs. On the flip side, however, it reduces table entries because organizations would not face arbitrary allocation limits on the number of hosts (EIDs) they could maintain in their AD. Interior routing would only need to scale to tens of thousands of nodes, a number already quite reasonable [21]. Further study is clearly required to answer this question, but we believe there is reason to be optimistic.

Update volume is a greater concern. Volume would likely increase linearly with the number of ADs, but a more serious possibility is that path length could grow if large ASes were split into several ADs. BGP's convergence time and update volume can grow with path length [16], so understanding the effects of such a change is a critical question to address. Fortunately, if update volume becomes unbearable, next-generation routing proposals such as HLP [28] successfully borrow techniques from link-state routing to improve the scalability and convergence time of wide-area routing.

6.2 Traffic Engineering

ISPs require control over routing to meet their traffic engineering goals; today, network operators typically manipulate traffic on the granularity of IP prefixes [6, 9]. AIP faces two challenges: ADs may be too large to permit effective traffic engineering (e.g., if ADs are like today's ASes), or ADs may be too small (e.g., if ADs were assigned at the granularity of today's prefixes). Which of these presents the major problem depends on how AIP is used; lacking a crystal ball, we outline solutions to each:

1. Aggregating ADs for TE: To perform traffic engineering at a more coarse granularity, AIP could use a LISP-like [5] tunneling approach, similar to proposals already under consideration by the IETF:

1. ADs prepend packets with a new address header containing an *outer* destination and source address. The outer addresses represent aggregates to ADs.
2. Operators configure IP layer TE policies using the less numerous outer addresses.
3. ADs discover a mapping from original destination to the outer tunnel address. These mappings must be maintained using an (unspecified) mechanism, but the process need not be complex because the mappings are relatively static.

2. De-aggregating ADs for finer-grained TE: The addresses contained inside an AD are also flat. To flexibly de-aggregate an AD, the routing announcements for the AD could be augmented with a "range" indicator that told other providers which EIDs to apply to this route. Such a mechanism would permit operators more fine-grained control at the cost of more expensive route lookups. Whether such an approach is better or worse than simply splitting an AD into smaller components is a question for future work.

6.3 Key Management

Keys will inevitably be compromised. The architecture must minimize both the chances of this event occurring and the pain involved in recovering from such an event.

Reducing the chance of compromise: We believe that in practice, domains would have a two-level key hierarchy. The self-generated domain key would be held very tightly and used to sign certificates for individual routers. This approach allows the administrators of an AD to choose between slightly increased complexity (though they still generate and maintain all of their keys completely independently of other entities) and resilience to key compromise. The question remains open whether this approach is better dealt with by revocation or short certificate lifetimes, who should maintain revocation lists, and so on.

Reducing the pain of recovery: Many questions remain in this area. Does this two-level key hierarchy lend itself better to revocation or to short certificate lifetimes? If an actual domain key is compromised, the domain *must* "re-number", since its very identity is compromised. What mechanisms (akin, perhaps, to today's DHCP and related protocols) can be used to make re-numbering an entire domain a relatively painless process? We note that solutions to this problem would benefit today's stub ASes and customers who switch providers. The problem, of course, touches upon router and peering configurations, host addressing, and name service, at a minimum. Because of the

complexity involved, not only would we need tools to deal with re-numbering, but careful thought must be put into the architectural dependencies to minimize the *number* and scope of the services that must be so updated.

7 Conclusion

It is high time that Internet entities be held accountable. No longer should hosts be able to forge addresses with impunity, nor attackers be able to hijack routes without fear of consequences. By basing Internet addressing on a simple principle—a flat, two-level hierarchy in which both address components are self-certifying—AIP brings new and needed accountability to the Internet architecture. While significant challenges must be surmounted to bring its ideas to fruition, we believe that the potential benefits of the proposal make it worthy of serious consideration by the community.

Acknowledgments

We thank Amar Phanishayee, Anirudh Ramachandran, Vijay Vasudevan, Mythili Vutukuru, Michael Walfish, and Dan Wendlandt for their contributions to some of the ideas in this paper. This work was supported by the National Science Foundation under awards CNS-0716278 and CNS-0520241.

References

[1] K. Argyraki, P. Maniatis, D. Cheriton, and S. Shenker. Providing packet obituaries. In *Proc. 3nd ACM Workshop on Hot Topics in Networks (Hotnets-III)*, Nov. 2004.

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

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

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

[5] D. Farinacci, V. Fuller, D. Oran, and D. Meyer. *Locator/ID Separation Protocol (LISP)*. Internet Engineering Task Force, July 2007. `http://www.ietf.org/internet-drafts/draft-farinacci-lisp-02.txt` Work in progress, expires January 18, 2008.

[6] N. Feamster, J. Borkenhagen, and J. Rexford. Guidelines for interdomain traffic engineering. *ACM Computer Communications Review*, 33(5), Oct. 2003.

[7] 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.

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

[9] B. Fortz and M. Thorup. Optimizing OSPF/IS-IS weights in a changing world. *IEEE JSAC*, 20(4):756–767, May 2002.

[10] 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.

[11] J. Ioannidis and S. M. Bellovin. Implementing Pushback: Router-Based Defense Against DDoS Attacks. In *Proc. Network and Distributed System Security Symposium (NDSS)*, Feb. 2002.

[12] 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.

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

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

[15] A. D. Keromytis, V. Misra, and D. Rubenstein. SOS: Secure overlay services. In *Proc. ACM SIGCOMM*, pages 61–72, Aug. 2002.

[16] C. Labovitz, A. Ahuja, R. Wattenhofer, and S. Venkatachary. The impact of Internet policy and topology on delayed routing convergence. In *Proc. IEEE INFOCOM*, Apr. 2001.

[17] J. Li, M. Krohn, D. Mazières, and D. Shasha. Secure untrusted data repository (SUNDR). In *Proc. 6th USENIX OSDI*, Dec. 2004.

[18] 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.

[19] R. Moskowitz and P. Nikander. *Host Identity Protocol (HIP) Architecture*. Internet Engineering Task Force, May 2006. RFC 4432.

[20] J. Moy. *OSPF Version 2*, Mar. 1994. RFC 1583.

[21] A. Myers, T. E. Ng, and H. Zhang. Rethining the service model: Scaling ethernet to a million nodes. In *Proc. 3nd ACM Workshop on Hot Topics in Networks (Hotnets-III)*, Nov. 2004.

[22] T. Okamoto and J. Stern. Almost uniform density of power residues and the provable security of ESIGN. In *ASIACRYPT*, pages 287–301, 2003.

[23] 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.

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

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

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

[27] I. Stoica, D. Adkins, S. Zhaung, S. Shenker, and S. Surana. Internet indirection infrastructure. In *Proc. ACM SIGCOMM*, pages 73–86, Aug. 2002.

[28] L. Subramanian, M. Caesar, C. T. Ee, M. Handley, M. Mao, S. Shenker, and I. Stoica. HLP: A next generation inter-domain routing protocol. In *Proc. ACM SIGCOMM*, Aug. 2005.

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

[30] 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.

[31] 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`.

[32] X. Yang, D. Wetherall, and T. Anderson. A DoS-limiting network architecture. In *Proc. ACM SIGCOMM*, Aug. 2005.