Accountability as a Service

Adam Bender, Neil Spring, Dave Levin, Bobby Bhattacharjee
University of Maryland, College Park
{bender, nspring, dml, bobby}@cs.umd.edu

Abstract

We propose that accountability be a first-class network service, independent of addressing and routing. We design a scheme for allowing *accountability services*, rather than connectivity-providing ISPs, to vouch for traffic, allowing victims to report abuse, filter abusive traffic, and isolate malicious senders. We discuss how accountability services may evolve, how they may facilitate new applications, and the implications of shifting the burden of network policing to a dedicated service.

1 Introduction

Accountability in a network is a means to identify the sources of traffic for two purposes: to selectively filter repeated unwanted, abusive, or non-compliant traffic from malicious sources on a per-destination basis, while permitting traffic from others to proceed, and to report and disconnect abusive machines before they attack others. IP addresses, if unforgeable, might serve to identify the sources of abuse, and Internet Service Providers (ISPs), if responsible for the actions of users of their networks, might disconnect compromised or abusive machines. In practice, neither is reliable.

The cooperation between and technical sophistication of operators of ISP networks has led to the assumption that ISPs actively police their networks: that all that is required to block misbehaving traffic is to trace it to the source ISP. Good operators react to reported abuse by disconnecting machines, repairing compromised hosts, and helping other operators do the same. Ingress filtering [11] and traceback [23, 3], if universally deployed, can simplify finding the machine that sent abusive traffic. However, the architectural ossification implied by this trust model, especially to mobility and in discouraging overlay routing, makes deployment costly. The timeliness with which misbehavior must be traced and the likelihood that a stepping stone [31] hides the true source make tracing to an ISP insufficient. Even if the source can be found, the source's ISP may not disconnect a paying customer based on a remote host's accusation, making traceback potentially ineffective.

The interests of ISPs include protecting the network, maximizing revenue, and reducing legal liability. They are not entirely aligned with the interests of endpoints—protecting an access link from spurious traffic, protecting a server or client from abuse, maximizing anonymity as a sender, maximizing accountability as a receiver. Host accountability is a tussle [5], one in which a specific, unsatisfactory design point has been implicitly chosen in the Internet architecture. It has neither sufficient accountability to discourage abuse nor sufficient anonymity to (by architectural means) support free speech and anonymous access to information. Buying connectivity from an ISP may not imply access to legal protection from that ISP as well.

Architectural support for accountability should provide both technical and social benefits. An ideal accountability architecture would support overlay networks, in which an application-layer relay may retransmit packets generated far away without accepting responsibility for their contents. It would permit anonymity, perhaps through onion routing [13] or through opaque addresses that do not identify the endpoint globally [12]. It would not restrict mobility, because source addresses within an ISP no longer imply that the ISP "vouches for" sent traffic. Users would know what privacy is provided. Servers would know how to identify, filter, and report repeat abusers.

In this paper, we argue for the separation of routing from accountability (which we term *accountability as a service*), and present a mechanism achieving this goal. A new set of services that we propose assume the role of "vouching for" the traffic generated by endpoints, allowing this endorsement to be well-defined, proxiable, and acceptable to both sender and receiver. This would supplant the implied, poorly defined endorsement of traffic by the ISP from which it originates, and move the tussle point from between the users and ISPs to between the users and accountability service(s). We believe that allowing a third party to vouch for traffic requires public keys and certificates, but that accountability as a service permits reasonably efficient implementation (for the benefit achieved). Each ISP retains a role in ensuring that outgoing packets are properly signed, but this filtering is deterministic and does not require inspecting packet payload [14]. Gateways in the network can be delegated the right to filter traffic on behalf of a destination as an optimization. Users and end-hosts decide which level of accountability in received traffic or anonymity in sent traffic is required.

We first describe related work that focuses primarily on ISP-level accountability and the tasks of a responsible network administrator in Section 2. We present an overview of an accountability service in Section 3, then describe two designs—one straightforward, one designed for cheap verification—for verifying accountable source identification

signatures in Section 4. We describe the interactions with applications and users, who will need to make decisions about their information, in Section 5. Finally, we discuss the implications of this architectural principle of separating routing from accountability in Section 6.

2 Related Work

Proposals for increasing accountability typically rely on the model that ISPs are responsible for policing the activities of their customers, investigating allegations of abuse and disconnecting guilty nodes, and that ISPs will address reports of abuse from their customers by installing filters upstream. Technical support for tracing abusive traffic to an ISP and selectively filtering abusive traffic fit well with this model; we approach accountability differently.

We define the role of ISP policing to include only verification that packets are correctly signed; approaches that trace packets to the source ISP remain useful, to ensure that signatures were checked. We draw heavily from Passports [15], allow a source ISP to cheaply sign packets for delivery through each AS in a BGP path. The passport indicates that the source address of the packet is correct but does not provide a means to map that address to a responsible sender without the cooperation of the origin ISP.

VPNs that traverse firewalls allow the VPN server to "vouch for" the traffic of an authorized client at the other end of an encrypted tunnel, placing a source address on each packet as if the client were part of the internal network. The effect is similar, in that responsibility for traffic is assumed by a remote server. However, because all traffic traverses the tunnel, privacy is limited. A firewall blocks all "unaccountable" traffic from entering the internal network from the outside Internet.

Unsolicited traffic, once its source is identified, can be blocked explicitly. Pi [28] and AITF [2] allow destinations to block incoming traffic that follows a recorded path (the path identifies the source). The victim authenticates the request to block traffic with a three way handshake to show that the request comes from the victim (or a node that can see its traffic). Simon et al. [22] define accountable traffic to be *identifiable* (using persistent identifiers) and *defensible* (identifiers are blockable). End-hosts can block traffic at the sender's AS, ASes must deploy ingress filtering and be able to map packets back to the sender.

Unsolicited traffic can also be blocked implicitly. TVA [30] prioritizes ongoing conversations over unsolicited messages. Routers insert authorization material into packets; if the destination returns that authorization material to the source, the source can send subsequent packets without risking rate limits. CAT [4] also allows senders to prove that subsequent packets are legitimate; an in-network "cookie box" provides flow cookies to senders that complete a three-way handshake.

The source discovered by traceback may not be the initial source of an attack. To skirt the accountability implied by sending directly to a victim, attackers use stepping stones [31]. Stepping stones can be detected [31, 24], attackers can attempt to evade detection, and some evasion methods can themselves be detected [9]. Promoting accountability to a first-class network operation may make stepping stones less common: client machines will likely expect inbound traffic to carry high-accountability, which might not be provided by a hacker.

Approaches to stop spoofed source addresses in electronic mail are similar. SPF [27] disallows spoofed source (email) addresses by checking that the relay has been approved to carry mail from the source address by being listed in DNS; this interferes with sending email through other relays. The Mail Abuse Prevention System (MAPS) [18] real-time blackhole list (RBL) is similar to an accountability service in our architecture, discovering abusive senders and publishing a list of senders who should be disconnected. Because SMTP connections require a TCP handshake, IP address spoofing is less of an issue.

3 The Accountability Service

The role of an accountability service is to provide *authenticated* clients with *identifiers* that can be used to *mark packets* accountable. Other clients of the service can *block* continued unwanted traffic, and *report* malicious, accountable packets (that they receive) to the service for further action. Accountability services may differentiate from each other by how much anonymity or accountability they provide and what they require from their clients: the number of identifiers given to each client, the intrusiveness of the proof of identity, whether proof of malice is required to disclose a client's identity, whether the accountability service holds an escrowed deposit for damages caused, the degree of trust an accountability service holds with clients and with servers, and other qualities. A client may choose among different accountability servers when sending or when receiving; we discuss the implications of such choice in Section 6.

We propose that accountability services will be designed as follows. The accountability service holds identities in escrow and typically reveals them only in cases of severe, proven abuse, or perhaps only by subpoena. The accountability service, not necessarily the ISP, vouches for the traffic of its clients; it collects reports of abuse and decides how to address them. Accountability services do not broker connections individually: accountability identifiers are independent of destination and may be reused. A sender may use different identifiers for different destinations to preserve anonymity. Accountability identifiers are proxiable: relayed packets may bear the identifiers of the sender, and relayed application-layer requests may include identifying headers (much like HTTP's proxied-for header, which identifies the original client). Receivers specify what accountability services they accept and what level of accountability they require. A victim can ask the network to filter traffic that has a specific identifier for any reason; the accountability service need not verify that the traffic was malicious or penalize the source.

As expected, our goals include incremental deployability, a compact representation of identifiers, and efficient verification. However, we do not intend network-layer accountability to replace application-layer authorization; although IP addresses have been used this way, e.g., for `rsh` and for Web access, we hope to not revisit those mistakes.

4 Design

We now describe a system to achieve these goals. We begin with a description of a straightforward signature-based approach. Then we present a service using lightweight cryptographic techniques to achieve a higher-performance solution comparable to, and building upon, recent proposals [30, 15, 2]. We describe how to contact an accountability service without accountability identifiers (bootstrapping) and how to report the error of missing or inadequate accountability identifiers (accountability faults).

4.1 A Straw-man Protocol: Signing Every Packet

Today, signed certificates bind keys to identities. When an online merchant wants to prove its identity to others, it first proves its identity to a trusted, well-known certificate authority (CA) which then issues a public key certificate. This certificate typically contains the name and public key of the merchant and is signed by the CA's private key. It can be verified by anyone who has the CA's public key and represents a statement that the CA vouches for the merchant's identity.

This expression of identity leads to a straightforward method of implementing accountability. An accountability service, A, acts as a CA, and provides a sender S with a key pair (s_{pub}, s_{priv}) and signed certificate $\mathsf{cert}_S = \{S, s_{pub}\}_{A_{priv}}$. In each packet, S includes a signed hash of the packet contents and cert_S. Anyone on the forwarding path, including the receiver, can verify that cert_S is valid certificate and that the packet hash was signed with s_{priv}. Any packet with an invalid certificate or a bad signature is dropped.

A receiver can block an abusive sender by filtering packets that carry the abusive sender's certificate. Certificates expire, and those of abusive clients would not be renewed (lest fewer receivers accept the accountability service), so filters need not last longer than certificates.

Although routers have the information to verify all signatures, the process is too expensive, so routers would likely leave checking to the receiver. The scheme is thus vulnerable to senders that use bogus certificates: the network wastes bandwidth and the receiver wastes computation to check bogus certificates, but learns nothing of the sender's identity.

4.2 An Efficient Protocol: Verifying Near the Source

We next compel ISPs to verify signatures near the source: a source using invalid signatures, violating the only rule of accountable traffic, would be filtered and disconnected. We rely on the first-hop ISP to verify certificates and signatures so that invalid ones are caught before they traverse the network. To ensure that routers participate, we make use of Passports [15], a method of authenticating the ISP from which a packet originated.

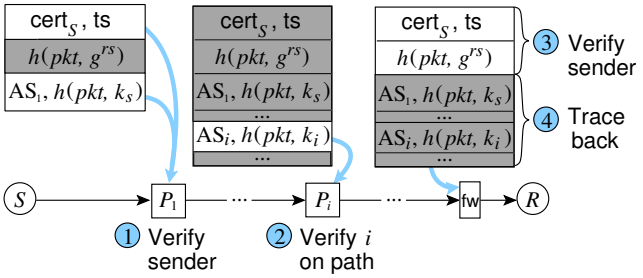

Figure 1: Ensuring the source ISP verifies certificates and enforcing that certificate holders compute valid signatures. Shaded regions represent pieces not verified at the next hop. A firewall operating on the receiver's behalf filters incoming traffic; the firewall may be colocated with the receiver or protect an access link.

In this protocol, sender S and receiver R agree to use accountability service A. A includes in its public key a group $\mathbb{G}$ and a generator g of $\mathbb{G}$. Each client C of A has as a private key integer c (lower case), public key g^c, and the certificate cert_C signed by A. When S, with public key g^s, wants to communicate with R, it must know R's public key, g^r. The sender and receiver create a shared Diffie-Hellman [8] key, g^{sr}, by raising the other's public key to their private key, obtaining $(g^s)^r = (g^r)^s$ in $\mathbb{G}$.

S's ISP, P_1, verifies cert_S (using A_{pub}) and that S knows s before forwarding, using the following procedure. If P_1 is also a customer of A, it will have a public key g^{p_1}, so S can compute a shared key $k_S = (g^{p_1})^s = (g^s)^{p_1}$. Otherwise, P_1 issues a challenge response protocol (over a secure, authenticated channel) by taking g from A_{pub}, choosing a random nonce n, and giving g^n to S. If S responds with $g^{ns} = (g^s)^n$, P_1 knows that S knows s and can set k_S. P_1 caches cert_S and k_S for fast verification in the future. S includes in each of its outgoing packets (1) cert_S (the certificate assigned by A), (2) a timestamp, used to prevent replay attacks, (3) a hash h_R of the packet contents, timestamp, cert_S, and the receiver's shared key ($h_R = h(pkt, ts, \mathsf{cert}_S, g^{rs})$), and (4) a hash h_1 of the packet contents and k_S ($h_1 = h(pkt, ts, \mathsf{cert}_S, k_S)$). We assume that, when used with a secret key, $h()$ provides the unforgeable property of a MAC. The ISP P_1 is expected to check that cert_S matches the cached value, that the timestamp is valid, and that $h_1' = h(pkt, ts, \mathsf{cert}_S, k_S)$ matches h_1.

The receiver validates cert_S to learn that A vouches for S. If cert_S is not filtered, the receiver computes the shared key g^{rs} and caches it with cert_S; further traffic from S can be accepted by verifying only that h_R was created using g^{rs}. As long as valid certificates appear in each packet, R can quickly distinguish good and bad senders.

To identify an ISP that does not check certificates in packets, R may trace packets it receives back to their source ISP using Passports [15]. P_1 arranges a shared key, k_i, between itself and each ISP P_i on the path to R. To do so, P_1 must have each P_i's public key, which is distributed in BGP. The public key infrastructure (PKI) used by Passports is separate from any accountability service's PKI. P_1 inserts into each packet

from S to R P_i's AS number and hash $h_i = h(pkt, ts, \mathsf{cert}_S, k_i)$. When P_i receives the packet, it checks the hash and drops the packet if it is invalid or missing. R may examine its incoming packets' list of ISPs to not only trace the packet back to P_1, but to present evidence of non-compliance to ISPs that presumably have a shared key with P_1. If R receives a packet with an invalid certificate, it can show to any P_i on the $S-R$ path a packet from S that was hashed by P_1 using the shared key of P_1 and P_i. P_i would be able to verify that cert_S is invalid (by using A_{pub}), thereby proving that P_1 did not check it. P_i could in turn issue an abuse report to P_1 or, in an extreme case, not accept any further traffic from P_1.

The Passport facilitates pushing filters on the path from S to R. R asks its ISP, P_n, to block traffic from cert_S on its behalf. P_n sends a "please block" message to a subset of the ISPs in a packet bearing cert_S. The message sent to P_i consists of cert_S, R's IP address, and the hash $h(R\text{'s IP}, g^s, k_{i,n})$ where $k_{i,n}$ is the shared Passports key between P_i and P_n. No host can forge a message that would block a sender's packets from reaching R without knowing $k_{i,n}$. Our scheme improves upon AITF [2] in that AITF allows anyone who can observe traffic from S to R to block that traffic and requires a 3-way handshake; in ours, only R's ISP may block traffic destined for R and only a single message is needed.

If blocking is not sufficient to stop abuse, R can present evidence of abuse to A. Depending on A's policy, it may reveal the identity of S, forbid S further use of the service, charge S a fine for abuse, or do nothing at all if the evidence is insufficient.

A combination of Bloom filters to catch short-term duplicates and timestamps to catch stored and replayed packets prevents replay attacks on both timescales. Not only can a malicious sender send traffic (that congests links, although it may not reach the receiver) until its certificate is revoked (which happens on a relatively long timescale), any node that observes packets from S to R can resend them to R. This duplication could cause a compliant sender to appear abusive. The network will need to filter duplicate packets, so that (1) links do not come under attack, and (2) innocent senders are not blamed for duplicated traffic. We make use of Passport's mechanism to prevent replay attacks by routers along the path. Routers use Bloom filters to catch short-term duplication, and fast rekeying of inter-ISP Passport keys, via hash chains, to prevent duplication across longer time scales. Bloom filters are refreshed when the keys are, to allow for smaller filters and reduce false positives. Timestamps prevent forgeries within an ISP. If hosts S and K share an ISP P_1, K could record S's packets and replay them, which might cause cert_S to be blocked, unless P_1 checks the timestamp.

This scheme has several attractive properties. Certificates are checked close to the source, allowing packets with bad certificates to be filtered before wasting additional network resources. The destination performs signature verification only for new certificates: the hash of g^{sr} is required in the packet to make subsequent verifications fast. ISPs can permit anything signed by a known-good certificate while blocking known-bad ones without having to block an entire neighboring ISP. This is the purpose of including certificates in packets. Each ISP need only store a few shared Passport keys; these can likely be cached for a long period of time, saving computation costs.

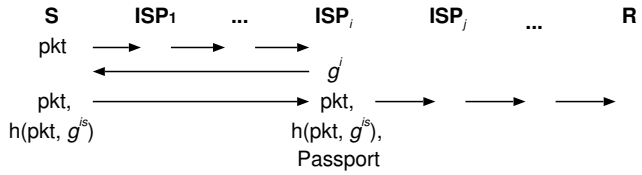

Figure 2: When S wants to send to R, and S's ISP does not support accountability, S must find the first ISP on the path to R that does, ISP_i. ISP_i then verifies S's signatures and inserts Passports.

The space required in each packet is significant, but not prohibitively large. The certificate occupies most of the space: elliptic curve public keys can be represented in 41 bytes and signatures in 40 [10]. The timestamp requires four bytes. Passports consume the rest: assuming 64-bit hash function output and 2-byte AS numbers, and that most Internet paths traverse no more than four ISPs [16], they require $(4+1)(8+2) = 50$ bytes. Because Passports cannot trace fragmented packets, we do not allow fragmentation.

To permit a sender in an ISP that does not support accountability to contact a receiver that requires accountability demands special attention for incremental deployability. As we show in Figure 2, S can first send a message with no accountability toward the destination, requesting the public key of the first ISP through which accountable messages can be routed. The first ISP, P_i, that implements this scheme (perhaps S's own ISP) returns its public key. S then requests P_i to act as its "first-hop accountability service" (to check cert_S and add the Passport hashes into the packet).

Accountability may be proxied (and thus used in overlay networks). S can request that entity S' act as a proxy for S, then arrange a shared key k' with the ISP of S' and give S' the accountability information that it needs to send packets through its ISP: g^s, cert_S, and $h(pkt, ts, \mathsf{cert}_S, k')$. ISPs may have incentive to offer accountability services to their customers as it would eliminate the necessary proxying and reduce accountability request messages. Also, peering agreements would influence ISPs; if an ISP is forced to do excessive proxying for its customer ISP, for example, it could impose penalties as defined in a service-level agreement.

4.3 Bootstrapping Accountability

Since an accountability server must receive packets that lack accountability identifiers, when clients request keys, it cannot rely on accountability to discourage abuse. If it is attacked, blocking all unaccountable traffic would render it useless. To solve this problem, we rely on traditional protective measures. First, an accountability service should be distributed and well provisioned. Second, its servers can be localized and firewalled, so that a dedicated server han-

dles requests for nearby ISPs only. Third, ISPs could be accountable for the traffic their customers send to accountability servers only, perhaps by placing their own identifiers on (possibly sanitized) requests to white-listed accountability servers.

ISPs might choose never to allow unaccountable traffic to exit the network. Such a policy would be expected from "free" or municipal wireless networks, who would want to provide connectivity, but not to be responsible for tracking users' activities. Such ISPs could allow access to white-listed accountability servers, run their own accountability services that provide basic identifiers to traverse their network, or proxy (and sanitize) accountability-setup requests to accountability servers.

4.4 The Accountability Fault

A router or firewall may receive a packet that lacks the "right" credentials. For example, the receiver might not support the accountability service the sender uses. We term such events *accountability faults*, alluding to the idea of page faults in virtual memory systems.

An accountability fault will result in a discarded packet, and may also generate an error message back to the source. This message is a reminder of which accountability services the receiver accepts; the sender can retransmit with a service the receiver supports. Accountability fault messages may also be used as a type of pushback [17] message; intermediate routers, upon receiving such a message, may discard messages that lack proper credentials or generate accountability fault messages of their own.

5 Who is Accountable?

Malicious traffic may be caused by a bug (the implementor is at fault), misconfiguration (the administrator), or abuse (the user): who "vouches for" the traffic? We believe the following rule of thumb applies: if an implementor believes his software or device cannot be abused, he may accept responsibility; else, if the application has no user, it is the administrator; else, it is the user.

Allowing the implementor to be accountable enables embedded devices to send accountable traffic even when they cannot acquire user-specific identifiers. The accountability services providing access to those devices would likely not be trusted by many destinations, if it was likely that the embedded devices can be hacked or implemented poorly [7]. For devices with too little power to provide accountability at all, a nearby relay or NAT may assume responsibility for (and check) the traffic they generate (receive). Today's model of network location as identity can be expressed.

To allow the user to act as a principal requires that operating systems ensure that each user's keys are isolated, that the same keys are used for the same destinations (preserving some anonymity), and that applications can use the user's keys. Pushing accountability to the end station requires conscious involvement of the user to decide how much personal information to disclose and how accountable incoming traffic must be. A user agent may aid this process by managing keys and preferences for identity disclosure to different servers.

A compromised endhost may be forced to account for malicious traffic. We believe that users are responsible for securing their machines, and that it is reasonable to isolate compromised hosts until they are cleansed. This isolation can be facilitated by accountability services, which can deny identifiers to a compromised machine even when its ISP takes no action. We expect that vendors of anti-virus and intrusion prevention software might provide a type of accountability service in which their clients would prove active use of current software as part of earning access. Such bundled intrusion prevention software may detect (signed, accountable) probes and report infection to an accountability service to isolate other compromised machines.

The direct implementation on received traffic may mirror a firewall policy in which the rules do not "accept" or "deny" all traffic to a port, but "accept if accountable." Hosts may use a function like `getpeername()` to collect the identifier associated with the remote endpoint and log accesses or responses.

6 Discussion

In this paper, we argue for accountability as a first-class network service, separate from routing and addressing. We believe that balancing the needs of anonymity and accountability should be the role of organizations that can make reasoned (and public) decisions about how and why to divulge the identity of a message sender.

Separating accountability from routing can foster new network services. The potential offered by anonymous network transmission [6, 13, 20] and the compromise to accountability required to use overlay networks [1, 21, 26] advance the need for accountability as a first-class network service. Rich overlay services are likely targets of abuse [19], and domain-level or IP traceback techniques that would point the finger at a specific host would only penalize overlay participants. Overlay packet forwarding for anonymity and performance can be permitted more easily when members of the overlay need not vouch for the traffic of their peers.

We do not address other facets of accountability in the network, for example, users assigning blame for poor performance to a specific provider or content providers holding users accountable for infringement. Separating routing from accountability may aid the deployment of source routing techniques that may make network service more competitive [5, 29]. We focus on partially-anonymous clients, but our model may be extended to contact partially-anonymous services through, for example, *i*3 [25].

We cannot predict the ecology of accountability services. They may be large or small, many or few, interested in protecting clients or in protecting servers. The resulting network may prioritize "more" accountable traffic over less. Some accountability services may be designed to supplant some uses of VPNs. The cost of such services may be expensive at first and be eventually subsidized (by advertising or

by services for whom accountability provides relief from attacks) or made free like email. Whether content providers or consumers are more likely to fund accountability services is unclear, as services act on behalf of both.

The social cost of accountability choice may be high in that the demands of service providers may reduce every user's expectation of anonymity, as cookies and targeted advertising do today. Attacks may encourage servers to require greater accountability in received packets. Some communication might be prevented: those unwilling to present sufficiently-identifying information might be unable to connect.

We expect that services with strong accountability requirements, customers that require protection for access links, and recent victims of denial of service attacks would drive early deployment. Accountability as a service can be useful with only one accountability-literate client, one accountability-dependent server, and one accountability service providing keys. Restated, this accountability architecture does not require global deployment to be effective. With accountability support in hosts and client-side NATs, networks could optimize by filtering closer to attack sources and accountability services could provide identities to more clients and servers. With sufficient deployment that accountability information replaces source addresses in logs, overlays and free wireless networks could carry traffic without enforcing acceptable usage policies.

In the Internet, ISPs can try to sanitize and filter bad traffic, but they are constrained in collaborating to prevent it [14]. We believe that it is in an ISP's interest to carry traffic "vouched for" by third parties because it may reduce their liability for misbehavior and their need to locally account for sent traffic. This is in contrast with deploying ingress or egress filtering—practices that are likely to make the ISP *more* accountable for bad traffic emanating from within, as identifying the ISP becomes easy while identifying a user remains difficult. By consistently defining verifiably accountable traffic and allowing endpoints to define end-to-end accountability requirements, the task of preventing misbehavior in the network might be made tractable.

References

[1] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. Resilient overlay networks. In *ACM SOSP*, 2001.

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

[3] S. M. Bellovin, M. Leech, and T. Taylor. ICMP traceback messages. Internet Draft: draft-ietf-itrace-04, 2003.

[4] M. Casado, *et al.* Cookies along trust-boundaries: Accurate and deployable flood protection. In *USENIX SRUTI*, 2006.

[5] D. D. Clark, J. Wroclawski, K. R. Sollins, and R. Braden. Tussle in cyberspace: defining tomorrow's Internet. In *ACM SIGCOMM*, 2002.

[6] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Int'l Workshop on Design Issues in Anonymity and Unobservability*, 2000.

[7] R. Clayton. The rising tide: DDoS from defective designs and defaults. In *USENIX SRUTI*, 2006.

[8] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[9] D. L. Donoho, *et al.* Multiscale stepping-stone detection. In *Fifth International Symposium on Recent Advances in Intrusion Detection*, 2002.

[10] E. C. Efstathiou, P. A. Frangoudis, , and G. C. Polyzos. Stimulating participation in wireless community networks. In *IEEE INFOCOM*, 2006.

[11] P. Ferguson and D. Senie. Network ingress filtering. IETF RFC-2827, 2000.

[12] P. Francis and R. Gummadi. IPNL: A NAT-extended Internet architecture. In *ACM SIGCOMM*, 2001.

[13] D. Goldschlag, M. Reed, and P. Syverson. Onion routing. *Communications of the ACM*, 42(2), 1999.

[14] S. C. Lee and C. Shields. Tracing the source of network attack: A technical, legal, and societal problem. In *IEEE Workshop on Information Assurance and Security*, 2001.

[15] X. Liu, X. Yang, D. Wetherall, and T. Anderson. Efficient and secure source authentication with packet passports. In *USENIX SRUTI*, 2006.

[16] D. Magoni and J.-J. Pansiot. Analysis of the autonomous system network topology. *ACM CCR*, 31(3):26–37, 2001.

[17] R. Mahajan, *et al.* Controlling high-bandwidth aggregates in the network. *ACM CCR*, 32(3):62–73, 2002.

[18] MAPS. http://www.mail-abuse.com/.

[19] V. S. Pai, *et al.* The dark side of the web: An open proxy's view. In *ACM HotNets*, 2003.

[20] M. K. Reiter and A. D. Rubin. Anonymous Web transactions with crowds. *Communications of the ACM*, 42(2):32–48, 1999.

[21] S. Savage, *et al.* The end-to-end effects of Internet path selection. In *ACM SIGCOMM*, 1999.

[22] D. R. Simon, S. Agarwal, and D. A. Maltz. AS-based accountability as a cost-effective DDoS defense. In *USENIX HotBots*, 2007.

[23] A. C. Snoeren, *et al.* Hash-based IP traceback. In *ACM SIGCOMM*, 2001.

[24] S. Staniford-Chen and L. T. Heberlein. Holding intruders accountable on the Internet. In *IEEE Symposium on Security and Privacy*, 1995.

[25] I. Stoica, *et al.* Internet indirection infrastructure. In *ACM SIGCOMM*, 2002.

[26] L. Subramanian, I. Stoica, H. Balakrishnan, and R. Katz. OverQoS: Offering internet QoS using overlays. In *ACM HotNets*, 2002.

[27] M. W. Wong and W. Schlitt. Sender policy framework (SPF) for authorizing use of domains in e-mail. IETF RFC-4408, 2006. Experimental.

[28] A. Yaar, A. Perrig, and D. Song. Pi: A path identification mechanism to defend against DDoS attacks. In *IEEE Symposium on Security and Privacy*, 2003.

[29] X. Yang. NIRA: A new Internet routing architecture. In *ACM SIGCOMM FDNA Workshop*, 2003.

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

[31] Y. Zhang and V. Paxson. Detecting stepping stones. In *9th USENIX Security Symposium*, 2000.