An Accountability Architecture for the Internet

Adam Bender Rob Sherwood Neil Spring Bobby Bhattacharjee
University of Maryland

ABSTRACT

Accountability in a network, defined as the ability to map packets in a network back to the principals responsible for their transmission, coupled with the capacity to block those principals, is vital to preventing misbehavior. Blocking malicious traffic in routers requires filters to distinguish malicious from valid traffic far from the victim. This architecture conflates the roles of ISPs as providers of connectivity (the pipe) and of accountability (stopping abuse). We believe that ISPs need not bear the responsibility for deciding the policy (whom to block) and implementing the mechanism (filters) of network layer accountability.

Towards this end, we present NA, an accountable Internet architecture. NA assumes the existence of trusted identity authorities, the ability to compute HMACs at intermediate routers and verify signatures at customer-edge routers, and the availability of servers to act as partially-trusted storage devices and representatives of an accountability provider.

With these features, we present protocol mechanisms that allow high performance verification of accountable traffic, traceback of unaccountable traffic, and the revocation of permission to connect.

1. INTRODUCTION

Misbehavior in the Internet can be attributed to two major factors: first, that attacks can be launched with relative anonymity through source spoofing or networks of compromised machines (botnets), and second, that it may not in a provider's interest to deny connectivity to a paying customer simply because some attack traffic leaves the network. Packet traceback [12, 14, 16] and reverse-path filtering to prevent spoofing [9, 13] attack the first problem, limiting the anonymity of attackers and allowing an abused destination to reliably accuse a sender of misbehavior. In each approach, however, the network providing connectivity to the source machine is expected to disconnect it (perhaps completely) to protect a distant destination.

As profit-driven companies, Internet service providers are not impartial: each has a financial interest in turning a blind eye to misbehavior by their own customers.[1] Rogue ISPs such as Atrivo [3] actively prosper from hosting services that are widely known to be malicious. for any protocol designed to find, block, and prevent network abuse is to *not rely on any ISP*. Restated, *providing connectivity must not imply accepting responsibility.*

We take this principle one step further; that a sender's ability to inject traffic into the network should be decoupled from his point of attachment. *Blocking* a misbehaving source should prevent that source from obtaining a new address or spoofing a different address to send traffic to the same destination. This suggests that identities, or more specifically blockable principals, should be issued as public keys from a globally-known, trusted authority. This trusted accountability provider (TAP) prevents arbitrary users from sending traffic and, more importantly, bars blocked subscribers from sending under a pseudonym. We generalize this design to support many TAPs, each with its own set of policies and underlying implementations, which agree on a protocol for validating packets in transit. That is, we attempt to design for the tussle [5], not for an outcome.

In this paper we present NA, a complete network accountability system in which:

1. An accountability provider is the sole trusted entity; all other participants (routers, accountability provider representatives, etc.) can be implicated if they misbehave. We ensure that its services need not be implemented centrally, allowing it to be made scalable and resilient to attack.

2. Endhosts can block principals from sending traffic to them. Blocked principals cannot send traffic to the blocking destination regardless of their point of attachment.

3. Blocking filter state is not stored at any router; a combination of Passport [9]-inspired tokens and distributed filter storage compels the source network to check for blocks.

4. A privacy certificate authority (CA) [15] authenticates temporary keys, rather than long-lived identifiers, for use in contacting destinations, preserving (some) privacy. Machines near senders guard the sender's long-lived identity, though we do not explicitly anonymize clients or servers as in SOS [6].

5. What constitutes abuse is determined by a destination alone, it need not be proven to an authority: if

[1]Protecting their customers from outside attack may be a separate service.

traffic is undesired, future traffic can be blocked. The sender can continue to send to other destinations.

6. Cryptographic operations can be largely avoided along the fast path through *waivers*, explicit statements that a server will not be implicated by a client, and *tokens* that enable traceback.

To implement this network accountability system, we designed, implemented, and evaluated protocol mechanisms for high-performance verification of accountable traffic without all-pairs shared keys, traceback of unaccountable traffic (improperly signed, but carried anyway) to a misbehaving inter-domain link, and revoking permission to connect without a separate, protected control channel.

The rest of the paper is structured as follows: we summarize related work in Section 2. In Section 3 we present the design of NA. We discuss attacks on NA in Section 4 and NA partial deployment approaches in Section 5. We evaluate NA overhead in Section 6 and conclude in Section 7.

2. RELATED WORK

NA allows receivers to block specific senders. Several recent proposals achieve similar goals using different mechanisms.

In TVA [17], senders initiate a connection by sending a *capability request* on a dedicated channel; this request is marked by each router along the way. If the traffic is desired by the receiver, the marked request is returned to the sender. The sender then includes this returned *capability* in subsequent packets. During an attack, packets without capabilities are preferentially dropped. Portcullis [11], an extension to TVA, uses sender-solved puzzles to protect the capability request channel. Unlike NA, TVA and Portcullis both pin the network path between the sender and the receiver (and require a symmetric path for optimal performance). NA relies on the TAP to issue global identities that enable receivers to block senders regardless of their network point of attachment; such a facility does not exist in TVA or Portcullis. Finally, NA requires only the TAP nodes be correct and trusted; all other entities may be corrupt. TVA and Portcullis require a trusted and correct transit infrastructure.

AIP [1] binds self-certifying addresses with network interfaces and assume that all interfaces have trusted hardware that obeys a "shut-off" packet sent by a receiver. Using these primitives, AIP builds an accountability infrastructure that receivers can use to stop abusive senders. Like NA, AIP provides receivers fine-grained control over which senders are blocked; unlike NA, the blocks in AIP are bound to network interfaces and not to packet-sending principals. NA relies on trusted authorities to certify network-layer-independent identities. In comparison, AIP relies on a pervasive trusted hardware deployment at end hosts. NA is designed as a shim layer between network (IP) and transport (TCP) and is agnostic of the network protocol, whereas AIP requires a new addressing scheme at the network layer.

Packet passports [9] is a system for unambiguously identifying a packet's source AS. This information can be used to track abusive senders. The packet passport system relies on a shared key between every pair of participating ASes (whether they are neighboring ASes or not). These keys are used to create HMACs (like the tokens of NA) that can eventually be used to identify the source AS of any packet. Passports require the predetermination of AS paths for every end-to-end route. The requirement of shared keys between every AS pair (packet passports) vs. pervasive trusted hardware (AIP) vs. trusted identity authorities in NA represent explorations in three starkly different points in the design space for network-layer accountability. NA minimizes the number of trusted entities in the system (only the TAP nodes are trusted) and is the only system that can withstand malicious transit routers and ASes.

Simon et al. [13] present a system where receivers can identify the source of traffic and block traffic from any source. Their scheme consists of per-customer ingress filtering at ISPs, and a trusted Filter Request Server (FRS) located in each ISP. To block a source, a receiver contacts the local FRS, which in turn contacts the FRS in the source's ISP. The source's FRS is responsible for filtering traffic from that source to the receiver. AITF [2] uses a mechanism similar to IP Record Route to mark the provenance of packets. When a receiver wants to block traffic that follows a certain path, the receiver asks its gateway to contact the sender's gateway, similar to how FRSes contact each other. In comparison to these systems, NA's notion of blocks transcend the sending principal's network point of attachment. Perhaps more significantly, these schemes hold the sender's AS responsible for traffic from the AS and for blocking abusive senders. In fact, AITF includes a provision for disconnecting an entire AS if blocks are not properly administered. In NA, destinations block individual sending principals; once blocked, these senders can no longer obtain necessary tokens to send packets to the blocked destination. The source AS is unaware of specific blocks, and instead simply verifies that only accountable packets depart its network.

Finally, IP traceback [12, 14, 16] is a mechanism that victims invoke to (partially) trace back the path an attack packet traversed. Traceback mechanisms require minimal changes to hosts and can potentially be used as a building block for an end-to-end accountability solution. The "accusation" protocol in NA uses tokens to traceback unaccountable packets. The NA traceback mechanism always isolates the malicious entity

that sourced the attack packets.

3. NA: DESIGN

The primary goal of NA is to ensure that only accountable packets may transit networks to destinations that have not blocked the sender. Accountable packets are those signed by a principal authorized by a trusted authority (*waivered* packets require no signatures, but carry proof that the destination expressly consented to receiving such packets). We term "unaccountable" those packets with an invalid signature and "legacy" those packets having no accountability material at all.

In NA, *principals send packets to hosts*. Each packet contains sufficient information for a destination host to block the sending principal (for any reason). NA provides the following property:

> Packets are forwarded to a destination only if the destination has not blocked the sending principal, regardless of the sender's network point of attachment.

Lesser goals that shape our design are as follows. We wish to distribute verifiable functions away from the trusted authority, to protect it from attack or overload with mundane requests; this leads to complexity but should provide resilience. We limit computation cost in transit networks by avoiding asymmetric cryptography in transit; this leads to larger packets. We avoid revealing user identifiers: even though IP addresses could be used to correlate accesses by the same user, we prefer that network-layer accountability not require revealing identity to destinations. We wish to allow return traffic to be sent cheaply; this allows servers and high-volume connections to avoid much of the NA overhead. Finally, we wish to support extremely many fine-grained blocks in the system, leading to a need to distribute the blocks across many servers.

We divide time into *epochs* on the order of minutes. All participants are synchronized to within an epoch; all certifications of public keys have an expiry time expressed as the expiration epoch. In our implementation, we use seconds since midnight UTC, January 1, 1970, as returned by `gettimeofday()`, divided by 512 to provide an easily computed, uniform epoch in 32 bits.

A sender may be able to continue sending packets to a destination even after the destination has blocked the sender. However, the "period of vulnerability" is bounded by the epoch. A blocked sender may also be able to send packets if it can find aid from a corrupt component; however, NA will expose the corrupt entity to keep principals blocked.

3.1 Components Overview

The NA architecture consists of five components which we describe in turn.

The TAP The TAP is the trusted accountability provider. Its role in the architecture is to certify and renew the keys of well-behaved, but untrusted, participants (each of which is described below). It must be able to accept proofs that a component misbehaved (signed statements that catch the component in a lie), deny certificate renewal to those participants, and not reissue new certificates to the same. All entities know the TAP's public key and can verify TAP signatures.

The TAP is the only trusted entity in the system. We assume that the TAP never divulges its private key and always discharges its protocol obligations correctly. The TAP may be implemented in a distributed manner with arbitrarily many replicas. These replicas need to synchronize only the current TAP public/private key pair and a list of principals whose keys are not to be renewed in the current epoch.

Packet-sending Principals (Users) A principal (or a user) $\mathcal{P}$ in NA is identified by a public key, $\mathcal{P}_{pk}$, which the TAP certifies (by signing) through an out-of-band mechanism. In Section 5, we define principals that forward legacy traffic (from existing machines, bearing no signatures) into the accountable network, likely with limitations. These *legacy gateways* appear to the architecture as ordinary users that can be blocked.

Autonomous Systems (AS) If the entire network were under the control of the same administrator, protecting against misbehavior would be simpler. Although no part of NA requires that the trust domains be split along autonomous system boundaries, we use the term AS because the idea connotes responsibility for routers and ownership of IP address ranges for hosts. The terms of pairwise connections between ASes are encoded in service level agreements, similarly, these pairwise connections may represent different levels of trust or different ways to address misbehavior.

Administrators of ASes have *speaks-for* keys, signed by the TAP, to express that the administrator is permitted to place blocks on behalf of specific destination addresses or to grant waivers that allow return traffic to be unsigned. These keys may be delegated, restricting sub-keys to more-specific address prefixes. The TAP binds a speaks-for key to a prefix by including the prefix its the signature of the key.

Neighboring ASes periodically exchange a symmetric key; border routers can create and verify message authentication codes (as realized in *tokens*, described below in Section 3.5) using this key.

An AS may permit hosts to exchange packets within the same AS without verification; one might allow traffic to NRs, to diagnostic services, or to bootstrapping (DHCP or DNS) servers, for example, without signatures. However, we expect that such legacy traffic would not be accepted by neighboring ASes. An AS *may* choose to transit, limit, or (likely with increased de-

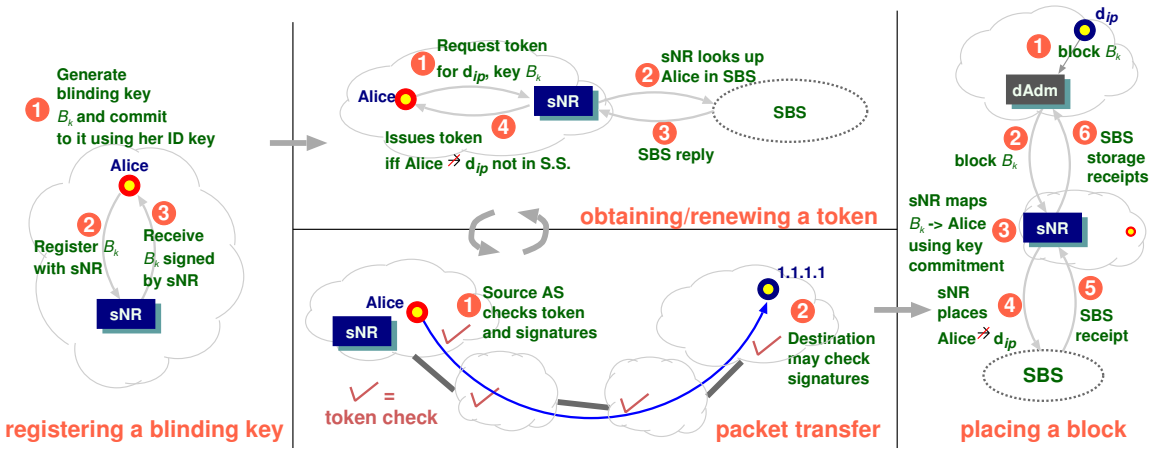

Figure 1: Life cycle of accountable packets. S.S. is the secure block storage mechanism.

ployment) outright reject legacy traffic, but *must not* transit unaccountable traffic.

Routers in an AS may be malicious or be compromised; we describe attacks by routers in Section 4.3.

NRs NA Representatives (NRs) are servers, distributed in the network, that acts as a privacy CA by certifying short-lived *blinding* keys for legitimate principals and generate initial *tokens* after checking for a block or validating a waiver. Blinding keys "hide" principal keys to provide a measure of unlinkability across connections. Tokens are an optimization that allow ASes to forgo asymmetric cryptographic operations.

Each NR has a key signed by the TAP. Each AS[2] contains at least one NR; like neighboring ASes, NRs periodically exchange a symmetric key with its host AS.

NRs may be malicious or compromised.

Secure storage The list of blocks—statements of the form "10.0/16 rejects packets from Principal A"—is maintained in a distributed, secure block storage (SBS) mechanism. We use the NeighborhoodWatch secure DHT [4], which nicely dovetails with NA. The NeighborhoodWatch DHT (NWDHT) is resilient to malicious nodes that drop or misroute queries. NWDHT provides security via a centralized authority, which we are aleady assuming (the TAP). Conversely, NA can be used to protect the NWDHT central authority, weakening NWDHT's assumption that the central authority be protected.

SBS nodes may be co-located with NRs or be independently provisioned. Each SBS node has a key signed by the TAP authorized it to act in that role.

3.2 An Accountable Packet

[2]As we shall see in Section 5, it is not strictly necessary for *each* AS to have an NR.

Accountable packets have a valid signature chain from the TAP to the packet: the TAP's signature of the NR's public key, the NR's signature of the blinding key, and the blinding key's signature of the packet contents. This chain make the packet accountable to the sender by providing a means to identify and block the sender. The sender is responsible for the packet, the NR is responsible for verifying the sender is not blocked, and the TAP is responsible for ensuring correct behavior of the NR. If the sender or NR is not behaving according to protocol, the destination can appeal to the next-highest authority present in the signature chain.

Verifying the entire signature chain of a packet in transit would be computationally infeasible. Thus, certain routers in every AS place tokens in the packet to signify that they have checked the signatures, or trust someone that has. An initial token, provided by the NR, indicates that the destination has not blocked the sender. The source AS is responsible for checking both the token and the signature chain. Subsequent tokens, added by each AS, indicate that the signatures verify—or the packet was received from another AS that claimed the signatures verify.

To further reduce the computational overhead of sending traffic, *waivered* traffic is exempted from signatures. If a sender S trusts return traffic from an intended destination D, S may give D permission to send return traffic without inserting *any* signatures into the packet. D's NR verifies that S has permitted unsigned return traffic before issuing an initial token to D.

3.3 Process Overview

We next outline how NA operates in normal operation, with misbehavior only by the source principal. We will then describe the steps in more detail. Assume that Alice is a principal, with public key $\mathcal{P}_{pk}$ certified by the

TAP, who wishes to talk to address d_{ip}; in turn, d_{ip} will wish to block Alice from sending further messages. Figure 1 illustrates this process and participants at a high level.

Registering a blinding key

1. Alice commits to a blinding key that she will later use to obtain tokens from a nearby "source" NR (sNR) (Section 3.4).

Obtaining an initial token

1. Alice contacts sNR with a registered blinding key and an intended destination (d_{ip}) (Section 3.5).

2. sNR examines the SBS to find any block that d_{ip}'s AS's administrator may have placed on Alice

3. If there is no block, sNR provides a token that enables Alice to send to d_{ip} (Section 3.5).

Constructing a waiver (optional)

1. Alice signs a message for the destination's NR that will instruct it to provide tokens to the destination without a lookup into the SBS (Section 3.7).

Packet Transfer

1a. Alice signs her outgoing packets with her blinding key and includes an initial token; a router in her AS checks signatures and the token. Or,

1b. For waivered packets, Alice hashes the waiver token with the message.

2. Each egress router places a token in the packet. Upstream ingress routers discard packets with invalid tokens. This process occurs on every peering link (Section 3.6).

Placing a block

1. The administrator of d_{ip} decides to block Alice. He places the block by sending the blinding key to sNR. (Section 3.8).

2. sNR places a block against $\mathcal{P}_{pk}$ (not the blinding key) into the SBS and returns a receipt to the administrator (Section 3.8).

Tokens are valid for only a single epoch. Therefore Alice must receive new tokens for every epoch in which she wishes to send. Once sNR receives a request to block Alice from sending to d_{ip}, sNR no longer gives new tokens to Alice (for sending messages to d_{ip}). Alice cannot send messages to d_{ip} by changing her network point-of-attachment since her (new) local NR will find the block in the SBS and not issue her an initial token.

In the rest of this section, we describe the messages exchanged for each of these sub-protocols. We denote a principal A's public and private key pair by A_{pk} and A_{sk} respectively, a message m signed using key k by $[m]_k$ and a message m encrypted using key k by $\{m\}_k$.

3.4 Registering a blinding key

Assume that Alice has generated a key pair ($\mathcal{P}_{pk}$, $\mathcal{P}_{sk}$), that the TAP has certified $\mathcal{P}_{pk}$ and that Alice wants to send messages to d_{ip}. Assume that sNR is within Alice's AS (AS_1) and that Alice's host is preconfigured with sNR's address. Alice generates a blinding key pair ($\mathcal{B}_{pk}$, $\mathcal{B}_{sk}$) and commits to it by signing $\mathcal{B}_{pk}$ with $\mathcal{P}_{sk}$. The format of Alice's commitment (Commit_{Alice}) is $\mathsf{Commit}_{Alice} = [\mathcal{B}_{pk}, \nu_{\mathcal{B}}]_{\mathcal{P}_{sk}}$ where $\nu_{\mathcal{B}}$ is the epoch until when Alice proposes to use this blinding key. The key registration process is as follows:

$Alice \mapsto \mathsf{sNR} : \mathsf{Commit}_{Alice}, [\mathcal{P}_{pk}]_{\mathsf{TAP}_{sk}}$; *registration request*
; *sNR checks that the TAP signed* $\mathcal{P}_{pk}$
; *sNR stores* $\mathcal{B}_{pk} \leftrightarrow \mathcal{P}_{pk}$
$\mathsf{sNR} \mapsto Alice : [\mathcal{B}_{pk}, \nu_{\mathcal{B}}]_{\mathsf{sNR}_{sk}}$; *signed blinding key*

Alice may register as many blinding keys as sNR will allow.

3.5 Obtaining an initial token

Initial tokens are bound to a blinding key, $\mathcal{B}_{pk}$, a destination, d_{ip}, the current epoch, c, and an issuing NR. Initial tokens make the statement "a valid $\mathcal{P}_{pk}$ registered $\mathcal{B}_{pk}$ and d_{ip}'s administrator has not blocked $\mathcal{P}_{pk}$." An initial token is required to send packets to a different domain and is checked by routers in Alice's domain.

When Alice wishes to communicate with d_{ip} she obtains an initial token as follows:

$Alice \mapsto \mathsf{sNR} : [\mathcal{B}_{pk}, \nu_{\mathcal{B}}]_{\mathsf{sNR}_{sk}}, \mathsf{d}_{ip}$; *token request*
; *sNR verifies its signature on* $\mathcal{B}_{pk}$
; *sNR verifies that* $c \leq \nu_{\mathcal{B}}$
; *sNR retrieves* $\mathcal{P}_{pk}$ *and checks the SBS*
; *to ensure Alice has not been blocked by* d_{ip}
$\mathsf{sNR} \mapsto Alice : \mathcal{T}_0$; *initial token*

The format of the initial token ($\mathcal{T}_0$) is:

$$H(K_{\mathsf{sNR}\leftrightarrow \mathrm{AS}_1}, \mathcal{B}_{pk}, \mathsf{d}_{ip}, c, \mathsf{sNR}_{pk}, \mathsf{sNR}_{ip})$$

where $K_{A\leftrightarrow B}$ is the shared symmetric key between A and B (A, B are neighboring ASes or an NR and the AS it is in). $H(\cdot)$ is a secure MAC, such as HMAC, with the first parameter as the secret key.

3.6 Packet Transfer

With an initial token, Alice can send a packet to d_{ip}. She signs her message with her blinding key and includes her initial token in the packet. Figure 2 shows the fields Alice embeds.

Forwarding an accountable packet consists of first, verifying the material within the packet, and second, appending the next token that the subsequent AS will verify. These two steps may be separated, so that packets are verified when they enter a network and given tokens just before they exit.

The first gateway to see the packet must verify the embedded signature chain from the TAP to the message: $[\mathsf{sNR}_{pk}, \mathsf{sNR}_{ip}, \nu_{\mathsf{sNR}}]_{\mathsf{TAP}_{sk}}$, $[\mathcal{B}_{pk}, \nu_{\mathcal{B}}]_{\mathsf{sNR}_{sk}}$, and $[m]_{\mathcal{B}_{sk}}$. It

IP Header	20
Version + Flags	2
Transport protocol	1
TAP index	2
sNR_{ip}	4
sNR_{pk}	14
$\mathcal{B}_{pk}$	14
$\left[\mathsf{sNR}_{pk}, \mathsf{sNR}_{ip}, \nu_{\mathsf{sNR}}\right]_{\mathsf{TAP}_{sk}}$	40
$[\mathcal{B}_{pk}, \nu_{\mathcal{B}}]_{\mathsf{sNR}_{sk}}$	28
$[m]_{\mathcal{B}_{sk}}$	28
ν_{sNR}	4
$\nu_{\mathcal{B}}$	4
c	4
$\mathcal{T}_0 = H(K_{\mathsf{sNR}\leftrightarrow \mathrm{AS}_1}, \mathcal{B}_{pk}, \mathsf{d}_{ip}, c, \mathsf{sNR}_{pk}, \mathsf{sNR}_{ip})$	4
$\mathcal{T}_1 = H(K_{\mathrm{AS}_1\leftrightarrow \mathrm{AS}_2}, \mathcal{B}_{pk}, \mathsf{d}_{ip}, c, H(m))$	4
$\mathcal{T}_2 = H(K_{\mathrm{AS}_2\leftrightarrow \mathrm{AS}_3}, \mathcal{B}_{pk}, \mathsf{d}_{ip}, c, H(m))$	4
$\cdots$	
$\mathcal{T}_{n-1} = H(K_{\mathrm{AS}_{n-1}\leftrightarrow \mathrm{AS}_n}, \mathcal{B}_{pk}, \mathsf{d}_{ip}, c, H(m))$	4
Transport (message m)	*var*

Figure 2: Accountable packet format. The rightmost column represents the number of bytes required for each field, when using cryptographic primitives described in Section 6.

IP Header	20
Version + Flags	2
Transport protocol	1
$\mathrm{HMAC}_{nonce}(m)$	4
$\mathcal{T}_0 = H(H(K_{\mathsf{sNR}\leftrightarrow \mathrm{AS}_1}, \mathsf{d}_{ip}, c), m)$	4
$\mathcal{T}_1 = H(K_{\mathrm{AS}_1\leftrightarrow \mathrm{AS}_2}, \mathcal{B}_{pk}, \mathsf{d}_{ip}, c, H(m))$	4
$\mathcal{T}_2 = H(K_{\mathrm{AS}_2\leftrightarrow \mathrm{AS}_3}, \mathcal{B}_{pk}, \mathsf{d}_{ip}, c, H(m))$	4
$\cdots$	
$\mathcal{T}_{n-1} = H(K_{\mathrm{AS}_{n-1}\leftrightarrow \mathrm{AS}_n}, \mathcal{B}_{pk}, \mathsf{d}_{ip}, c, H(m))$	4
Transport (message m)	*var*

Figure 3: Waivered return packet format.

then verifies the initial token $\mathcal{T}_0$ by checking that (1) sNR_{pk} and sNR_{ip} belong to the same sNR, (2) $\mathcal{T}_0$ was created for $\mathcal{B}_{pk}$, and (3) the MAC computed with the $K_{\mathsf{sNR}\leftrightarrow \mathrm{AS}_1}$ that it shares with sNR is correct. If any check fails the packet is dropped. These steps are unique to the first gateway because it is responsible for validating the entirety of the packet; later gateways check only that the packet was validated by a (transitively) trusted neighbor.

Before the packet leaves the network, a router stamps the next token on the packet. This token conveys that the packet is valid: that the signatures and previous token have been checked and that the next domain should transit it, even if the token's creator has not directly verified the signatures in the packet. The format of the next (and any subsequent) token is: $\mathcal{T}_1 = H(K_{\mathrm{AS}_1\leftrightarrow \mathrm{AS}_2}, \mathcal{B}_{pk}, \mathsf{d}_{ip}, c, H(m))$. The packet arrives at the destination with a chain of tokens that can be used to diagnose in-network misbehavior (Section 4).

The use of tokens does not require shared keys between every AS pair or the predetermination of AS paths, as in Passports [9].

3.7 Waivers

High-volume servers and power-constrained devices may not be able to sign each packet they transmit. NA allows hosts to state with a waiver that unsigned return traffic will not be blocked. Hosts must possess or have access to a valid speaks-for keypair (SF_{pk}, SF_{sk}) delegated to their prefix; we expect the key to be delegated specifically for one address for a short duration, but a waiver may be generated by a service, such as the NR.

We assume that Alice can transmit a connection nonce n to Bob in secret; perhaps by encrypting it using Bob's well-known or stored public key, in-band in an established secure transport protocol atop accountable packets, or by other means. Alice then sends a waiver ω of the form:

$$\mathsf{SF}_{pk}, [\mathsf{SF}_{sk}]_{\mathsf{TAP}_{pk}}, Alice_{ip}, e, [H(n), Alice_{ip}, e]_{\mathsf{SF}_{sk}}$$

where e is the epoch until which any node possessing n may send waivered packets to $Alice_{ip}$. The waiver is only valid if SF_{pk} is signed; if SF_{sk} is delegated, the certificate chain from the TAP is included in the waiver. Alice sends ω to Bob.

Bob verifies the signature of the hash of the nonce and forwards ω to its NR. The NR verifies that $Alice_{ip}$ created a waiver allowing Bob to send to Alice and grants a *secret* waiver token for sending to $Alice_{ip}$ without querying the SBS. The format of a waiver token is $H(K_{\mathsf{sNR}\leftrightarrow \mathrm{AS}_1}, Alice_{ip}, c)$. However, because this token is not bound to a principal, only to the holder of a nonce, it could be abused if eavesdropped. To keep this waiver token secret and prevent replay, NR encrypts it before transferring it to Bob. The waiver token is not included directly in the packet, but hashed with the message.

Bob does not include any signatures in packets for Alice; instead, Bob includes a HMAC of the packet computed using Alice's nonce as the key. Bob's AS allows waivered packets from Bob to depart without alarm. The format of the waivered packet is shown in Figure 3.

Figure 4 presents a state transition diagram showing how a sender transitions to and from being able to send waivered traffic. The sender traverses the path through obtaining an initial token to being able to send accountable packets. It may receive a waiver to send outgoing packets, $\overrightarrow{\omega}$; this causes it to obtain a waiver token from the NR and then begin sending waivered packets. From either sending state, the sender may add the $\overleftarrow{\omega}$ to permit unsigned incoming traffic. If a token or waiver is not renewed, the sender reverts to a less- authorized state.

3.8 Placing a Block

Suppose the owner of d_{ip} wishes to block Alice. he forwards one of Alice's packets to his administrator (say dAdm). Given a packet from Alice, dAdm can extract

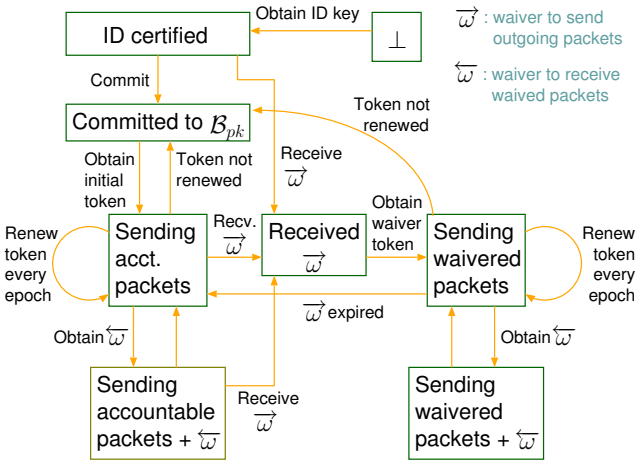

Figure 4: NA sender states and transitions.

sNR's address and Alice's blinding key. Assume that dAdm possesses the speaks-for key corresponding to address block `1.1/16`. dAdm places the block:

$$\mathsf{BlockReq}_{1.1/16} = \text{“1.1/16”}, \mathsf{dAdm}_{pk}, \mathcal{B}_{pk}, \mathsf{d}_{ip}, type, \\ [\text{“1.1/16”}, \mathsf{dAdm}_{pk}]_{\mathsf{TAP}_{sk}}, [\mathcal{B}_{pk}, \mathsf{d}_{ip}, type]_{\mathsf{dAdm}_{sk}}$$

where $type \in \{$“new”, “republish”$\}$, as follows:

$\mathsf{dAdm} \mapsto \mathsf{sNR} : \mathsf{BlockReq}_{1.1/16}$ *; block request*

; sNR extracts Alice's $\mathcal{P}_{pk}$ and places a block in the SBS
; The SBS nodes return k signed storage receipts (r_i).

$\mathsf{sNR} \mapsto \mathsf{dAdm} : r_1, ...r_k, \{\mathsf{Commit}_{Alice}\}_{\mathsf{NR}_{pk}}$ *; block receipt*

The NR in the destination network, dNR, may grant tokens without consulting the SBS for the purpose of sending block messages to other NRs. This permits sending block requests while overloaded. dNR must inspect the combination of the block request and the packet headers to ensure that the request is not spurious: for example, that $\mathcal{B}_{pk}$ followed a chain from the TAP and that the destination for the token is sNR_{ip}.

Depending on the implementation, the SBS may not permanently store blocks. A dAdm that insists upon permanent blocks may republish them before they expire. To facilitate republishing, the sNR returns an encryption of Alice's public key. A republished block request includes this encryption to sNR, who decrypts it to find Alice's commitment. After Alice's original commitment expires, sNR may expunge it; it is the responsibility of dAdm to keep this state.

To block further waivered traffic, the victim must both block the principal used to solicit a waiver, if applicable, and refuse to renew the waiver.

3.9 Multiple TAPs

Our description focused on a single TAP. We expect many TAPs to be simultaneously deployed and for destinations to accept packets signed with identities issued by TAPs of their choosing. We assume that each TAP has a globally-unique index to be carried in all accountable packets and used to identify TAP_{pk}.

Senders must know which TAPs a destination honors, and NRs issue tokens to a sender only if the sender has a key signed by a TAP that the destination accepts.

Administrators use SF_{sk} to sign a list of mappings from address prefix to TAPs they accept. This mapping is public and can be disseminated through DNS.

4. ATTACKS

In the previous sections, we have described the default operation of NA, without failures or node corruption. In this section, we show how NA detects misbehavior and eventually evicts malicious nodes.

We classify attacks into three categories based on how attackers are identified:

NA: The protocol messages in NA include sufficient information to unambiguously implicate corrupt nodes.

Local: The NA protocol can resolve the misbehavior to two neighboring entities, either of which may be corrupt. However, the “good” entity knows the corrupt neighbor and modify its trust policy accordingly.

TAP: Some attacks require the assistance of a TAP node for resolution. Victims provide the messages that indicate a corrupt entity to the TAP, which can verify the deceit, decide to not recertify the node for future epochs, and communicate the decision to replica TAPs.

Table 1 matches the most prominent attacks NA components may attempt with the mechanism that would isolate the attacker. In the rest of this section, we describe selected attack resolution mechanisms in detail.

4.1 Source-Generated Attacks

Source floods NA enables destinations to explicitly block source principals. If a source floods a destination domain with accountable traffic (i.e., packets with valid blinding and NR keys), the victim network can block the source. We ensure that our block request message (from the dAdm) fits into a single Internet MTU IP datagram and does not require an acknowledgment for the block to be activated. Thus, the block protocol will be able to block the source in the next epoch, even if the source can saturate the victim's access link.

If the source sends unaccountable traffic (i.e. traffic with an invalid signature), then the source's domain—which is responsible for verifying each signature—will discard the packet.

If the source sends legacy traffic (without an accountability header), then the traffic may be forwarded or discarded as described in Section 5.

SBS storage exhaustion A malicious sender may try to overload the SBS by attacking many destinations,

Attack Description	Resolution
Sources may	
Flood destination⋆	NA: Victim places a block; tokens not granted
Change network point of attachment	NA: Block found in SBS from any location
Replay old tokens	NA: Tokens expire after one epoch
Use tokens to send to arbitrary destination	NA: Tokens bound to destination addresses
Exhaust SBS storage by incurring many blocks⋆	NA: NR rate-limits tokens to blocked senders
Falsely accuse waivered traffic of being unaccountable	Local: Destination can prove waiver is valid and accusation is false
NRs may	
Fabricate blinding keys; issue tokens to unregistered users⋆	NA: NR cannot produce a commitment from a principal to that key; the SBS will not return a receipt
Flood destination⋆	TAP: Administrator can expose NR to the TAP
Not place blocks; ignore block request⋆	TAP: Victim NR receives no receipt, redirects request to the TAP
Not lookup blocks before issuing tokens; issue tokens to a blocked sender⋆	TAP: SBS does not return receipts, reports duplicate block to the TAP
Routers may	
Not check signatures⋆	Local: Tokens will lead back to this network
Flood using bogus tokens	NA: Neighbors will discard packets with invalid tokens
Flood using bogus source	Local: Cannot create accountable traffic; tokens will lead back to this network
Replay packets⋆	Local: Neighbors install Bloom filters
Fabricate upstream tokens⋆	Local: Fabricator and neighbor network accuse each other
Not check upstream tokens	Local: If an invalid token is accepted, network will not deflect blame

Table 1: NA attacks and resolution. Items with a ⋆ are explained in Section 4.

causing the SBS to hold an increasing amount of block information We expect TAPs to restrict the number of distinct simultaneous blocks on a principal, after which the principal will not be issued new tokens, and may be excluded from further participation in the system. Corrupt colluding ASes could try to get an honest principal (h) blocked by issuing blocks on h. However, they could only do so if h had sent packets to each of these ASes. Any reasonable limit on the number of blocks (say 20-50) would ensure that honest principals are not susceptible to this attack.

4.2 Corrupt NRs

Flooding Like any sender, a corrupt NR may try to flood a destination or a TAP node. If the NR node sends packets using a blinding key registered to NR_{pk}, the blinding key would be blocked as if it were a normal sender. If the misbehaving NR does not honor the block, the administrator who blocked the NR does not receive receipts and forwards the request to the TAP, who eventually evicts the NR. If the NR does return receipts, but continues to send traffic, the administrator presents the TAP with the returned receipts and an example of the traffic that violates the receipts. The TAP evicts the NR.

An NR is unable to send packets using a registered blinding key (without the corresponding $\mathcal{B}_{sk}$, it is unable to sign packets). A corrupt NR can never implicate an honest user of misbehavior.

However, a corrupt NR may generate valid tokens for a blinding key that does not map to any registered user and use these tokens to attack a destination. All signatures in the packet verify. The victim administrator tries to block the source principal, but the block request does not succeed since the corrupt source NR is not able to produce a valid commitment (which the SBS requires in order to produce a receipt). The destination administrator fails to receive receipts from the source NR and complains to a TAP node. The TAP node challenges the source NR to produce a valid blinding key commitment, at which point the source NR will be exposed.

Ignoring block requests If an NR ignores block requests (or is unable to return the necessary number of receipts), the victim administrator must appeal to the TAP. The TAP will challenge the source NR, and eventually ensures that the block is placed (or evicts the NR).

Forgoing lookups or issuing tokens to a blocked sender A source and an NR may collude. The NR could issue a token allowing a sender (using a different blinding key) to send packets to a destination that has previously blocked the sender. NRs that forgo SBS lookups may do the same. In this case, the destination will presumably block the sender a second time. When the second block is published to the SBS, it will observe that the destination has blocked $\mathcal{P}_{pk}$ twice—an indication that the NR misbehaved. The SBS will not return a receipt, and may alert the TAP.

4.3 Corrupt Routers

Corrupt routers may fabricate packets with (bogus) accountability headers, resulting in packets with an in-

valid signature chain—preventing the destination from blocking the sending principal—to arrive at a destination. Similarly, routers may not check packet signatures, allowing unaccountable traffic to enter the network. Additionally, a corrupt router may simply replay valid packets to flood the victim.

Generating or forwarding invalid packets (Accusation Protocol) If a victim receives an unaccountable packet, it must rely on the tokens to isolate the faulty node. The victim's administrator inspects the last token in the packet and determines which of its upstream ASes the packet came from (by determining which shared key was used to check the token). The victim administrator then "accuses" the upstream AS of sending an invalid packet. The upstream AS (say AS_1) verifies the last token, and tries to map the second-to-last token to one of *its* upstream neighbors. If this process is successful (and the second-to-last token maps to AS_2), AS_1 "accuses" AS_2. This process recurses until the packet traverses the last "good" AS (say AS_g) into the AS that generated it (say AS_b).

Note that all that is required to verify a token is the blinding key, destination IP, epoch the packet was created, the hash of the packet contents, and the token itself. Thus accusation messages (containing the full token chain), like block requests, fit into a single packet.

When the original packet was generated, the corrupt router in AS_b may have fabricated all upstream tokens, in which case the AS_b administrator will have no other upstream AS to blame. In the worst case, the AS_b router may have been able to add one "valid" upstream token, since it may know the secret shared with the upstream AS.

In the first case, either the AS_b administrator will locate and expunge the corrupt router (and stop the attack), or the attacks will continue. If the attacks continue, each AS will continue to receive accusations from downstream routers and AS_g must apply some local policy. This policy could include technical solutions (e.g. check signatures on incoming packets) or monetary remedies (e.g. the AS peering SLA could state that the offending AS will pay a small sum for each unaccountable packet that it transits).

Suppose the AS chooses to validate signatures if the accusations continue. This will enable AS_g to immediately isolate AS_b (since AS_g will immediately know that AS_b is transiting unaccountable packets). AS_g will not forward these packets, stopping the flow of accusations. If AS_b continues to source unaccountable packets, AS_g may take further action.

If the SLA between AS_g and AS_b includes the monetary provision, then AS_b will have to pay for each unaccountable packet. AS_g may have also had to pay its downstream AS (after all, AS_g did transit an unaccountable packet); however, in this case, AS_g recoups its cost because of the payment from AS_b.

If the router in AS_b added a "valid" token for a previous hop AS (since the neighboring ASes share a secret key), AS_b will be able to "blame" a good upstream AS (say AS_{up}). However, AS_{up} will not be able to map the packet to any of *its* upstream neighbors. If the AS_{up} administrator can immediately assert that its routers have not been corrupted, it can isolate AS_b.

However, it is often difficult for a network administrator to certify that none of the hosts or routers in its AS has been compromised. In this case, AS_{up}'s administrator can locally audit the router between itself and AS_b, ensure it is not compromised, and verify the signatures of all packets traversing the peering link. If the accusations continue, then AS_b is unambiguously implicated, and AS_{up} may choose to renegotiate the peering.

Verifying signatures on every packet may be prohibitively expensive on high speed links. AS_{up} can implicate AS_b without verifying packet signatures. Once the AS_{up} administrator verifies that its peering router with AS_b (say R) is not compromised, it generates a temporary symmetric key (k), and instructs R to add *two* tokens on each outgoing packet. The first outgoing token is computed using k (known only to R) and the second token is the regular token computed using the key shared between AS_{up} and AS_b. If subsequent accusations from AS_b includes a token signed with k, then some host or router within AS_{up} is compromised, and the AS_{up} administrator has to audit its internal routers. However, if AS_{up} is not compromised, then the accusation packets from AS_b will not contain a token signed with k (since AS_b does not know k). However, all packets transited from AS_{up} included a token signed with k; this enables AS_{up} to unambiguously implicate AS_b.

Replaying valid packets A router can replay a fixed packet only until the next epoch, when it will be dropped by subsequent domains. However, this does not solve router replay in general. If routers persist in replaying accountable packets, Bloom filters can be installed in upstream domains, as in Passports [9]. When the Bloom filter detects a replay, the duplicate packet is stored. Further duplicates are compared against stored packets to ensure that the match was not a false positive. The filters can be refreshed every epoch, limiting the storage needed.

4.4 Attacks using waivered packets

Assume Alice sent a waiver allowing Bob to send unsigned packets to Alice. When Alice receives an unsigned packet, she can use the HMAC to assert that she had explicitly permitted the sender to send unsigned packets. If the HMAC in the packet is corrupt (or not present), Alice may use the usual "accusation" protocol to stop/isolate the sender.

If Alice were malicious, she may try to evict Bob (who

is good) by initiating the accusation protocol on packets that include a proper HMAC. The tokens in the packet would eventually terminate at Bob, who can produce Alice's waiver explicitly allowing Bob to send unsigned packets. Bob does not have to reveal his private key to decrypt the nonce in order to prove that Alice sent a waiver; he only has to produce the nonce and show that Alice signed a hash of the nonce. Bob now accuses Alice (tracing the packet tokens *forward*) of misbehavior. Bob's accusation includes Alice's waiver, her accusation message, and the nonce *in cleartext*:

$$\mathsf{SF}_{pk}, [\mathsf{SF}_{sk}]_{\mathsf{TAP}_{pk}}, \text{“accuse”}, nonce$$

Each AS on the forward path can now independently check that (1) Alice did allow Bob to send unsigned messages, (2) check that Alice signed the nonce and (3) that Bob had included a proper HMAC on the return packet. This evidence is sufficient for Alice's AS to isolate Alice's host (or for Alice's AS's upstream to isolate Alice's AS).

Bob's AS does not check signatures on waivered packets. However, this does not allow any new attacks. If Bob sends malicious traffic to a destination that has given a waiver to Bob, the victim domain will not renew Bob's waiver, and the NR in Bob's domain will stop issuing tokens to Bob when the waiver expires.

5. PARTIAL DEPLOYMENT

NA does not require global deployment. Partial deployment solutions depend on whether the original sender is NA-aware and whether there are legacy transit domains between accountable domains. We assume that all senders and destinations in an accountable domain are NA-aware; however, NA-aware principals may send packets from legacy domains.

Two new entities interact with legacy ASes. An *accountability gateway* enables NA-aware senders in legacy domains to send traffic to accountable domains. A *legacy gateway* forwards sanitized legacy traffic onto accountable domains.

Sending accountable traffic from legacy domains Accountable senders in legacy domains may wish to send traffic to domains that only accept accountable traffic. The sender's traffic is routed through an *accountability gateway*, NR_{gw}.

Accountability gateways have a TAP-signed NR key that they use to issue tokens to senders that have committed to a blinding key and have not been blocked, much like a normal NR. Unlike a normal NR, accountability gateways issue tokens to senders in legacy ASes; these senders must tunnel accountable packets to the accountability gateway's AS to use the token.

The blocking procedure is the same: should a receiver decide to block Alice, it follows the standard blocking protocol by sending a block request to NR_{gw}, who publishes the block in the SBS.

Key Size	Sign	Verify
112 bits	$\mu = 0.91$, $\sigma = 0.03$	$\mu = 1.11$, $\sigma = 0.01$
160 bits	$\mu = 1.77$, $\sigma = 0.01$	$\mu = 2.07$, $\sigma = 0.02$

Table 2: Times in milliseconds to perform each operation on a 3.4 GHz Intel Pentium 4 CPU. μ is average, σ is standard deviation.

Transiting through legacy domains Accountable ASes can tunnel over legacy ASes to extend the reach of deployed, accountable “islands”, much like the MBone and 6Bone. The tunnel endpoints exchange symmetric keys and tokens created with these keys span the tunneled “link”.

It may be the case that an egress router, r, in an accountable domain cannot forward an accountable packet to an accountable domain, for instance if it cannot find a tunnel that reduces the distance to the packet's destination or if r determines that the destination is in an adjacent legacy domain. In this case, r removes the accountability header from outgoing packets, changes the IP protocol field to the protocol field in the accountability header, recomputes the IP header checksum, and forwards the resultant legacy packet.

Legacy senders Legacy gateways allow legacy senders to send packets to accountable ASes. A legacy gateway ($\mathcal{L}$) possesses a $\mathcal{P}_{pk}$ from the TAP. $\mathcal{L}$ accepts legacy traffic, creates and commits to a blinding key corresponding to the source IP address, and forwards the packets using the usual NA protocol. In effect, $\mathcal{L}$ takes responsibility for the legacy traffic by signing it. Destinations that block $\mathcal{L}$ no longer receive any legacy traffic through this gateway. Like any user $\mathcal{P}_{pk}$, if sufficient destinations block $\mathcal{L}$, its key may no longer be renewed by the TAP.

6. EVALUATION

We have implemented the TAP and a NA router. We present microbenchmarks for each type of overhead from our implementation. The TAP is implemented in Ruby, using SWIG wrappers to OpenSSL [10]. We have augmented the Click router [7] to perform NA token manipulations and signature verification.

We use the OpenSSL implementation of ECDSA [8] for signatures and SHA-1 HMAC for tokens. ECDSA affords short public keys and signatures without expensive bi-linear pairing operations. We use 160-bit keys for TAP_{pk}, $\mathcal{P}_{pk}$, and speaks-for keys, and 112-bit keys for NR_{pk} and $\mathcal{B}_{pk}$ keys. (As the latter two types are refreshed relatively often; their keylength can be shorter.)

We present an evaluation of the overhead for different NA sub-protocols. NA overheads can be partitioned into

	Operation	Actor	Frequency	112-bit Signatures	Verifications 112-bit	Verifications 160-bit	Local RTT	Remote RTT	Header size	Msg. size
1	Send acc. packet	Sender	Per packet	1	·	·	·	·	185	*var.*
2	Verify acc. packet	Router		·	2	1	Local operation			
3	Commit to blinding key	Sender	As needed	1	·	·	1	·	20	100
4	Verify commitment	sNR		1	1	1	Local operation			
5	Request (initial) token	Sender	Once per epoch	·	·	·	1	·	20	50
6	Return (initial) token	sNR		·	[0..1]	·	1	·	20	4
7	Create a waiver	Admin	Per waivered connection	1	·	·	Local operation			
8	Return a waiver	Admin		·	·	·	1	·	20	76
9	Send a waiver	Sender		·	·	·	·	1	185	76
10	Check + register waiver	Server		·	·	2	1	·	20	76
11	Send waivered packets	Server	Per packet	·	·	·	·	·	48	*var.*
12	Create block request	Admin	Per block	1	2	1	1	·	185	124
13	Create accusation pkt	Admin	As needed	·	·	·	1	·	185	48
14	Forward accusation pkt	AS		·	·	·	1	·	185	48

Table 3: Overhead of NA sub-protocols. Header size reflects packets with five tokens. but y in the worst case. All signatures and verifications for the same operation can be performed in parallel.

five broad categories:

Accountable traffic: Accountable traffic incurs processing overheads for signatures and verifications, latency overheads for block lookups, storage overheads for storing commitments, and packet overheads for storing an accountability header.

Traffic with waivers: Once waivers are issued, NA operations require no asymmetric cryptography. The primary overhead is in constructing and checking HMACs.

NR and gateway: NRs and legacy gateways must store blinding key mappings, publish blocks to the SBS, and verify SBS receipts.

Administrator: Administrators sign block requests, create waivers, and store and verify SBS receipts.

TAP node: TAP nodes sign keys for principals and NRs. They resolve disputes and evict misbehaving NRs.

The primary computational overhead in NA comes from signature creation, signature verification and HMAC operations. The time required for the signature computations on a typical workstation is shown in Table 2. The HMAC operations require less than 0.02 ms for 1500 byte packets.

Table 3 shows the number of different cryptographic operations required for NA component protocols, the number of local and remote round trips each protocol incurs, and the sizes of the protocol messages. We observe the following from Table 3:

Line 2 One router in the source AS must verify two short signatures and one long signature for accountable packets. These verifications can be parallelized and two of them may be cached.

Line 3 Principals can pre-commit to blinding keys and use them as necessary. There is no need to commit to a blinding key for each new connection.

Line 7 Creating a waiver requires two signature operations, only one of which must be performed at connection setup time. The nonces can be generated and signed offline.

Line 11 Once a waiver is verified and a token obtained, no more asymmetric operations are required.

Line 12 If a packet can be verified (three verification operations) and the destination wishes to block the sender, it needs to create one signature with its speaks-for key. Block requests fit in a single datagram.

Line 13-14 If a destination receives a packet with an invalid signature or an invalid nonce, it invokes the accusation protocol. The accusation packets also fit into a single datagram, and require no signatures beyond the accountability header. Each successive upstream AS must check their token (one HMAC) and then compare the previous token against all keys shared with their other AS neighbors (one HMAC operation per neighbor).

The compilation of overheads in Table 3 suggests that NA can be deployed in the wide-area using current hardware. In the remainder of this section, we use the analysis in Table 3, connection-level trace data from access links at different institutions, and the microbenchmarks from our implementation to extrapolate the performance of a hypothetical NA deployment.

The local network administration at the authors' institution manually blacklists IP addresses by installing a firewall rule for persistent attackers. Between August 2007 and December 2007, 2,736 hosts were blacklisted (approximately 15 per day). The overhead of this level of blacklisting is negligible.

Next, we consider an extreme case where a single victim AS is synchronously attacked by a 10,000 node botnet, where each bot is from a different AS. This is the

worst case attack since the victim AS cannot simply block corrupt sNRs, it must install a new block per attacking principal.

Under attack, the administrator verifies three signatures and signs one message (plus the packet). The total CPU time per block is less than 7 ms; 10,000 block requests can be generated in 70 seconds. Each block request is 309 bytes in size, so an administrator with a 1 Mbps connection would require 25 seconds to contact all of the attacker's sNRs. Since block generation and deliver can happen in parallel, administrators require approximately 70 seconds to block 10,000 attackers in different domains. This is less than an epoch length; by the next epoch, the flood will be blocked.

7. CONCLUSION

NA is a network-layer accountability architecture that explicitly decouples connectivity from responsibility. NA's security is grounded in trusted identity certification authorities (TAPs). NA allows destinations to specify per-source principal blocks, which apply regardless of the principal's network attachment point. NA does not mandate local policy, e.g., each AS is free to choose how it classifies misbehavior, and can withstand the corruption of all non-TAP components.

Extrapolated numbers from our implementation of NA show that computationally, it is feasible to implement NA with current hardware. We have strived to make the trade-offs in NA explicit and we expect the point in design space that NA explores to be both viable and important.

8. REFERENCES

[1] D. Andersen, H. Balakrishnan, N. Feamster, T. Koponen, D. Moon, and S. Shenker. Accountable internet protocol (AIP). In *SIGCOMM*, 2008.

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

[3] J. Armin. Atrivo - cyber crime USA, 2008. http://hostexploit.com/downloads/Atrivo%20white%20paper%20090308ad.pdf.

[4] A. Bender, R. Sherwood, D. Monner, N. Goergen, N. Spring, and B. Bhattacharjee. Fighting spam with the NeighborhoodWatch DHT. In *INFOCOM*, 2009.

[5] D. D. Clark, J. Wroclawski, K. R. Sollins, and R. Braden. Tussle in Cyberspace: Defining Tomorrow's Internet. *IEEE/ACM Transactions on Networking*, 13(3):462–475, June 2005.

[6] A. D. Keromytis, V. Misra, and D. Rubenstein. SOS: Secure overlay services. In *SIGCOMM*, 2002.

[7] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The Click modular router. *ACM Transactions on Computer Systems*, 18(3):263–297, August 2000.

[8] J. López and R. Dahab. An overview of elliptic curve cryptography. Technical report, State University of Campinas, 2000.

[9] X. Lui, A. Li, X. Yang, and D. Wetherall. Passport: Secure and adoptable source authentication. In *NSDI*, 2008.

[10] The OpenSSL project. http://www.openssl.org.

[11] B. Parno, D. Wendlandt, E. Shi, A. Perrig, B. Maggs, and Y.-C. Hu. Portcullis: Protecting connection setup from denial-of-capability attacks. In *SIGCOMM*, 2007.

[12] S. Savage, D. Wetherall, A. R. Karlin, and T. Anderson. Practical network support for IP traceback. In *SIGCOMM*, 2000.

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

[14] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, S. T. Kent, and W. T. Strayer. Hash-based IP traceback. In *SIGCOMM*, 2001.

[15] Trusted Computing Group. TCG trusted network connect: TNC architecture for interoperability, 2008.

[16] A. Yaar, A. Perrig, and D. Song. Pi: A path identication mechanism to defend against DDoS attacks. In *Security and Privacy*, 2003.

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