Source Accountability with Domain-brokered Privacy

Taeho Lee, Christos Pappas, David Barrera, Pawel Szalachowski, Adrian Perrig

ETH Zürich
{kthlee, pappasch, david.barrera, psz, adrian.perrig}@inf.ethz.ch

ABSTRACT

In an ideal Internet, every packet would be attributable to its sender, while host identities and transmitted content would remain private. Designing such a network is challenging because source accountability and communication privacy are typically viewed as conflicting properties. In this paper, we propose an architecture that guarantees source accountability and privacy-preserving communication by enlisting ISPs as accountability agents and privacy brokers. While ISPs can link every packet that originates from their network to their customers, customer identity remains unknown to the rest of the Internet. In our architecture, network communication is based on Ephemeral Identifiers (EphIDs)—cryptographic tokens that can be linked to a source only by the source's ISP. We demonstrate that EphIDs can be generated and processed efficiently, and we analyze the practical considerations for deployment.

1. INTRODUCTION

The commercialization of the Internet and its integral role in our daily lives have spawned a debate on privacy and accountability—a long-standing discussion about two properties that are typically considered conflicting. Unfortunately, today's Internet does not provide native support for either. We propose an architecture that resolves the accountability-privacy tussle and guarantees network-level *source accountability* without forfeiting end-to-end *communication privacy*.

On one end of the spectrum, source accountability protects the integrity of the source's identity and holds the source responsible for any traffic that it originates. The lack of source accountability has become a Pandora's box for Internet security. Attackers spoof their addresses and launch massive reflection attacks exhausting the available network resources. IP spoofing makes it impossible to identify the actual attacker and renders traffic filtering ineffective, not to mention the collateral damage when incorrectly blocking benign hosts.

On the other end of the spectrum is privacy. Recent revelations of pervasive monitoring and mass surveillance [1–3] have increased users' interest in communication privacy. Users know that their identities and network traffic are being systematically collected by state-level entities. The lack of native support for private communication in the Internet forces users to rely on overlay networks and specialized applications to obtain privacy guarantees [4, 5]. These solutions are complex to install and manage, and degrade application performance.

To date, the research community has mainly investigated approaches that favor either privacy or accountability, typically offering one at the expense of the other. To our knowledge, the Accountable and Private Internet Protocol (APIP) is the main proposal that has aimed to find a balance between the two properties at the network layer [6]. However, the privacy guarantees are constrained to source anonymity; data privacy is not addressed but delegated to conventional protocols, such as IPsec [7] and its key-exchange protocol (IKE [8]) that in themselves do not explicitly address a critical problem: certificate management (e.g., issuance, revocation) at Internet-scale.

In this paper, we propose an Accountable and Private Network Architecture (APNA) that provides source accountability guarantees *and* privacy-preserving communication. Our notion of communication privacy covers host privacy (for the source and destination) and data privacy—host privacy means that the identity of the host (e.g., IP address) remains private and data privacy means that the transmitted data remains secret from unintended recipients.

To provide such properties, we enlist Internet Service Providers (ISPs) as a fundamental component of our architecture for several reasons. First, we build on past efforts to hold Autonomous Systems (ASes) accountable for malicious traffic generated within their domain [9, 10]. Second, we believe ISPs have business incentives to provide privacy features to their customers, especially in light of recent revelations regarding global surveillance. In APNA, ISPs facilitate connection establishment between communicating hosts, but traffic encryption is still performed directly by communica-

CoNEXT '16, December 12–15, 2016, Irvine, CA, USA.

 ISBN 978-1-4503-4292-6/16/12…$15.00

DOI: http://dx.doi.org/10.1145/2999572.2999581

tion endpoints, keeping communication content hidden even from their ISPs.

In our scheme, network communication is based on Ephemeral IDentifiers (EphIDs) instead of long-lived network addresses, such as IP addresses. ASes issue EphIDs and assign them to their customer hosts as tokens of approval for communication. EphIDs are designed to mask the host address in the network, providing host privacy, while still functioning as a return address. In addition, EphIDs are bound to short-lived and domain-certified public/private key pairs. These keys are used by hosts to mutually authenticate each other and to negotiate a shared secret key, which allows native payload secrecy through network-layer traffic encryption.

The privacy architecture proposed in this paper, which establishes shared keys based on EphIDs, by default encrypts all payload data. Pervasive encryption frustrates large-scale surveillance by obfuscating all communicated content. Moreover, the encryption scheme provides Perfect Forward Secrecy (PFS) such that an adversary that obtains all long-term keys cannot decrypt the content of previous communication sessions.

EphIDs are cryptographically linked to the identity of a host and serve as accountability units. ISPs issue and assign EphIDs only to their authenticated customers, thus bootstrapping source accountability. We argue that ISPs are the natural accountability agents in today's Internet since they already know the identities of their customers. Furthermore, we describe a shutoff protocol [11], which is a common security mechanism relying on source accountability. A complaining destination host instructs an ISP to block outgoing traffic from a customer-host that is associated with an EphID. The accountable identifiers allow an ISP to first verify that a customer has indeed sent traffic to a certain destination and then to prohibit any further communication.

Contributions: This paper proposes a cohesive architecture, the Accountable and Private Network Architecture (APNA), that simultaneously guarantees accountability and protects privacy by involving ASes as accountability agents and privacy brokers. In particular, APNA achieves the following properties:

- Source accountability by linking every packet in the network to its originating source.
- Host privacy by hiding the host's identity from every entity except the host's AS.
- Data privacy by supporting network-layer encryption with perfect forward secrecy.
- Support for a shutoff protocol that terminates unwanted communication sessions.

2. PROBLEM DEFINITION

Our goal is to design a network architecture that simultaneously supports source accountability while preserving communication privacy. This section describes the necessary requirements to realize these seemingly conflicting goals, the security properties we strive to achieve, and the adversary model we consider.

2.1 Source Accountability

Source accountability refers to an unforgeable link between the identity of a sender host and the sent packet. Thus, accountability ensures that a source cannot deny having sent a packet and a host cannot be falsely accused of having sent a packet which it did not send.

Achieving source accountability in practice translates to two fundamental requirements. First, a strong notion of host identity is necessary so that hosts cannot create multiple identities nor impersonate other hosts. Second, a link between the source's identity and all of its traffic must be established. This link must be established (or at least confirmed) by a third-party (e.g., source AS) that is not the sender itself, since senders themselves can be malicious. To this end, the third party must observe all of the sender's traffic such that every packet in the network can be linked to a specific sender.

Adversary Model: The goal of the adversary is to break source accountability by creating a packet that is attributed to someone else in the network. We assume that the adversary can reside in multiple ASes and that she can see all packets within those ASes. Specifically, the adversary can eavesdrop on all control and data messages in the network, but cannot compromise the secret keys of the ASes that it resides in.

2.2 Communication Privacy

Our first goal with respect to privacy at the network layer is host privacy. To achieve host privacy, the identity of a host must be hidden from any other host in the source AS that is not in the same broadcast domain as the host (e.g., on the same WiFi network or LAN segment), any transit network that forwards traffic, as well as the destination AS (including the communication peer). A host cannot hide from its AS, since the AS knows the identity and network attachment point of every customer; and, a host cannot hide from other hosts in the same broadcast domain, since the layer-2 address is visible. We address host privacy at the network layer, which means that network-layer headers should not leak identity information. A host's identity may still leak at higher layers (e.g., HTTP cookies); however, these aspects are out of scope for this paper.

In addition, our notion of host privacy includes sender-flow unlinkability [12]: simply by observing packet contents (both headers and payloads) of any number of flows originating from the same AS, the source(s) of the flows are no more and no less related after the observation than they were before the observation.

Our second goal is data privacy through pervasive end-to-end encryption. Transmitted data should be hidden from unintended recipients, including the source and destination ASes. To this end, the architecture must natively (i.e., without relying on upper-layer protocols) provide secure key establishment between hosts and protection against Man-in-the-Middle (MitM) attacks.

Moreover, our notion of data privacy includes perfect forward secrecy (PFS): disclosure of long-term secret keying

material does not compromise the secrecy of exchanged keys from past sessions and thus data privacy of prior communication sessions is guaranteed [13, p. 496].

Adversary Model: Breaking *host privacy* means that an adversary can determine the identity of a sender, or can determine if two flows from the same source AS originate from the same host. We assume that the adversary can control any entity in the Internet except for the source host, hosts that are in the same broadcast domain as the source host, and the source AS itself. The adversary can observe packet headers and content, but we do not consider timing analysis techniques, such as inter-packet arrival times.

We argue that the architecture should provide only the basic building blocks to achieve host privacy at the network layer; stronger privacy guarantees (e.g., resiliency against timing analysis) should be provided by protocols at a higher layer (e.g., transport protocol). For instance, a transport protocol could implement a packet scheduling algorithm that alters timing between packets to mitigate traffic identification based on inter-packet timing analysis [14–17]. Our argument is grounded by the fact that strong privacy guarantees often come at the expense of network performance, and not every user (or application) requires strong privacy guarantees. Hence, protocols that offer stronger privacy guarantees are left to upper layers so that users can choose the appropriate protocol based on their requirements.

An adversary can try to compromise *data privacy* by decrypting the communication content exchanged between two hosts. To this end, we assume that the adversary can control any entity in the Internet except for the two communicating hosts and one of the two ASes that the hosts reside in.

2.3 Additional Goals

Shutoff Functionality: An accountability architecture can provide security mechanisms that build on top of accountable addresses. A shutoff mechanism is commonly used to terminate any active communication session flagged for misbehavior. The architecture must ensure that the shutoff mechanism does not create other attack vectors, such as denial-of-service through non-permitted shutoff requests.

Support for Network Feedback: The architecture must allow the network to communicate back to the source without revealing the identity of the source. Feedback from the network is crucial for network-diagnostic and management tools, such as ICMP.

3. APNA OVERVIEW

This section describes the components of our Accountable and Private Network Architecture (APNA), beginning with the role of the ASes (Section 3.1), followed by the use of ephemeral identifiers (Section 3.2), and ending with an end-to-end communication example (Section 3.3).

3.1 Role of ASes

In APNA, ASes act both as accountability agents and as privacy brokers due to their position in the network. Since ASes already know the identity and the physical attachment point of their customers, they naturally act as *accountability agents*. At the same time, ASes mask their customers' identities from all other entities, and thus act as *host-privacy brokers*. In addition, ASes certify their customer-related information (e.g., public keys), which is then used to generate keys for pervasive data encryption at the network layer; thus ASes act as *data-privacy brokers*.

Accountability Functions: As an accountability agent, the AS performs the following functions.

First, the AS creates a strong notion of host identity. To this end, the AS ensures that subscribers do not create and use multiple unauthorized identities for their communication. ASes already authenticate their customers and are thus selected as accountability agents.

Second, the AS creates a link between the identity of the source and the sent packet. To this end, the AS can store every packet or insert a cryptographic mark into every packet. Regardless of the implementation, the AS is on the forwarding path of all the traffic originating from its customers and is therefore selected to establish this link. Using any other third party, which is not on the traffic path, as an accountability agent would require additional mechanisms to report every packet to the third party [6].

Third, the AS realizes the shutoff functionality by accepting (and validating) shutoff requests and blocking the corresponding flows. An AS is in a strategic position to block malicious traffic since it is close to the source and can stop traffic before it leaves its network.

Privacy Functions: As a privacy broker, the AS performs the following functions.

First, the AS issues an Ephemeral IDentifier (EphID) that a host uses to mask his identity by using it as the source address. This identifier serves as a privacy-preserving return address and thus does not break bidirectional communication. However, EphIDs must be bound to specific hosts; and, since ASes know the identities of the hosts, they are well suited to perform this binding and act as host-privacy brokers. We provide more details on EphIDs in the following section.

Second, the AS acts as a certificate issuer, certifying that a public key indeed belongs to a host in the AS's network. More specifically, the AS certifies the binding between an ephemeral identifier that is issued to a host and a public key that is bound to the identifier. Hence, the AS becomes a data-privacy broker without revealing the identity of its customers.

3.2 Ephemeral IDs

At the heart of our proposal is the use of ephemeral identifiers instead of addresses. An EphID is an identifier associated with the identity of a host, yet it does not leak identity information. Since ASes know the identities of their customers, issuing EphIDs to their connected hosts enables the hosts to hide their identity without sacrificing accountability.

EphID as an Accountability Unit: As an accountability unit, an EphID is an authorization token for communication that is issued by the AS to its customer hosts. Issuing these

tokens requires strong host authentication: the host must first prove its identity to the AS and only then EphIDs can be issued.

In APNA, a host is represented to its AS through a Host Identifier (HID). An HID could be a hash of the host's public key or a number that is assigned by the AS to the host (e.g., IPv4 address). We do not specify how an AS assigns HIDs, but require that HIDs be unique within the AS's boundary. There can be multiple EphIDs that are associated with an HID, and the EphIDs are cryptographically bound to the HID such that only the host's AS can determine the binding. Furthermore, an EphID serves as the accountability unit for shutoff requests. A shutoff request against an EphID terminates all flows of the host that use that EphID as the source identifier. In other words, flows with the same source EphID are fate-sharing with respect to the shutoff protocol. Blacklisting source EphIDs instead of source and destination EphID pairs forces hosts to carefully manage their pool of assigned EphIDs.

EphID as a Privacy Unit: The EphID has two roles as a privacy unit: it hides the identity of a host and provides a tool to achieve sender-flow unlinkability. An EphID is meaningful only to the issuing AS and opaque to all other parties. It reveals no information about the host's identity to other hosts inside the same AS nor to the peer host that the host is communicating with.

EphIDs alone are insufficient for routing packets to a destination, since location information is missing; and, in APNA, the location information is provided at the granularity of ASes. Hence, a host is fully addressed by an AS Identifier (AID) and EphID tuple (i.e., AID:EphID) where the AID identifies the AS in which the host resides (e.g., Autonomous System Number), and the EphID is the ephemeral identifier issued to the host by the corresponding AS. Hence, the only leaked information is the AS where the host resides; the host's anonymity set becomes the size of the AS in terms of number of hosts.

In addition, decoupling the identity from the address provides a means to achieve sender-flow unlinkability. A host can be issued multiple EphIDs and can use them at will, e.g., a single EphID for all flows or a different EphID for every flow. We do not impose any requirements on how EphIDs are used. We discuss different granularities of EphIDs in Section 8.1.

3.3 Communication Example

We describe the high-level workflow for communication between two hosts (Figure 1). The protocol details are provided in Section 4.

An AS needs to manage its hosts; issue and manage EphIDs; and authenticate packets that its hosts send. For these tasks, the following logical entities are present in every AS:

- **Registry Service (RS)**: authenticates and bootstraps hosts in the AS.
- **Management Service (MS)**: issues EphIDs to the hosts.
- **Border Router (BR)**: handles incoming and outgoing packets based on the AID:EphID tuple.
- **Accountability Agent (AA)**: handles shutoff requests against the hosts in the AS.

In Figure 1, a host in AS_A initiates communication with a host in AS_B. Communication proceeds in four steps:

1. **Host Bootstrapping:** the host authenticates to the RS of its AS and receives bootstrapping information.
2. **EphID Issuance:** the host contacts the MS of its AS to obtain an EphID.
3. **Connection Establishment:** the hosts know each other's AID:EphID identifiers and establish a shared key that will be used for network-layer data encryption. The shared key is derived from public keys that are associated with the EphIDs. In Section 7.1, we describe how hosts can obtain the necessary communication information through DNS.
4. **Encrypted Communication:** the hosts proceed with the actual communication by using their AID:EphID tuples instead of network addresses and by encrypting every packet with their shared symmetric key.

4. APNA PROTOCOL DETAILS

To construct a lightweight and efficient architecture, APNA is built under consideration of the following design choices:

- Symmetric encryption is used to cryptographically link EphIDs with HIDs; this allows an AS to efficiently obtain the HID from the EphID without a mapping table, which can be large.
- Proof of sending a packet is embedded in the packet, avoiding large amounts of stored state at ASes.
- Forwarding devices perform only symmetric cryptographic operations, guaranteeing high forwarding performance.

We begin by stating our assumptions and proceed with the details of the steps that are shown in our communication example (Section 3.3). Table 1 summarizes the notation we use throughout the protocol description.

4.1 Assumptions

- *We assume that the cryptographic primitives we use are secure.* For instance, we assume that authenticated encryption is used for encrypting data communication, securing data communication against chosen-ciphertext attacks (i.e., CCA-secure). We also require that the generation of EphIDs to be CCA-secure; in Section 5.1.1, we describe an efficient CCA-secure encryption scheme for generating EphIDs.
- *Participating parties can retrieve and verify the public keys of ASes.* For example, a scheme such as RPKI [18] can be used to verify the public keys of the corresponding ASes. For simplicity, ASes use the same public/private key pairs for 1) signing messages, and 2) key exchanges. In a real-world deployment, these two keys would be different, and the key used for signing messages would be registered with RPKI.

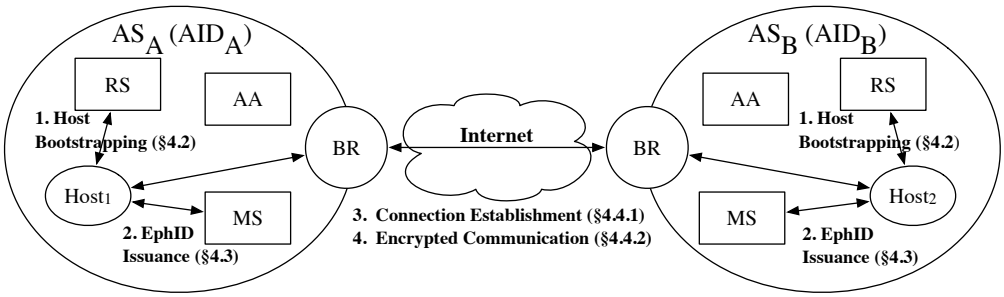


Figure 1: An end-to-end communication example.

Table 1: Notation.

k_{A_i}	Symmetric key of AS_i that is known among the infrastructure (e.g., routers, RS, MS, AA).
$k_{H_iA_i}$	Symmetric key shared between host H_i and its AS (AS_i).
$k_{E_iE_j}$	Symmetric key generated for the EphID pair E_i and E_j.
HID_i	Host identifier (HID) assigned to host H_i.
$EphID_h$	An EphID issued to host H.
C_{EphID_i}	Certificate for EphID $EphID_i$.
K_E^+, K_E^-	Public, private key of entity E for both DH Exchange and Digital Signatures.
$MAC_k(M)$	Message M and MAC of M using symmetric key k.
$\{M\}_{K^-}$	Message M and Signature of M using private-key K^-.
$E_k(M)$	Symmetric encryption of M with key k.
$E_k^{-1}(C)$	Symmetric decryption of C with key k.

- *Hosts do not use connection sharing devices (e.g., NAT).* We discuss how to relax this assumption in Section 7.2.

4.2 Host Bootstrapping

A host authenticates to its AS using a well-established authentication protocol [19, 20]. For example, when a user subscribes to an Internet provider, the provider creates the authentication credentials and these credentials are preconfigured into an Internet access device (e.g., cable or DSL modem). The device performs the authentication protocol when the user connects it to the network.

The host (or his access device) creates a symmetric key k_{HA} that serves two purposes: encrypting EphID request and reply messages (Section 4.3), and authenticating every packet that the host transmits to the network.[1] During the authentication procedure, the host securely sends k_{HA} to the RS by encrypting it using AS's public key (K_{AS}^+).

Once the host has successfully authenticated, the Registry Service (RS) of the AS performs the bootstrapping procedure (Figure 2). During this procedure, the host receives information about its AS's services that are necessary to (later) establish communication sessions; and to support these communication sessions, the infrastructure of the AS gets updated with the host's information. We require that all bootstrapping messages are authenticated in order to avoid modifications en route.

First, the RS creates a *control* EphID ($EphID_h^{ctrl}$) for the host. Control EphIDs are used to access AS's internal services, e.g., to request data-plane EphIDs from a Management Service (MS). Both control and data-plane EphIDs are constructed identically so that all communication is based on EphIDs. However, control EphIDs have longer lifetime (e.g., DHCP lease time) than data-plane EphIDs. In addition, control EphIDs cannot be used for data communication—unlike data-plane EphIDs, certificates are not issued for control EphIDs (see Section 4.3). We use the term EphIDs to refer to the data-plane EphIDs.

The RS returns the following information to the host: the control EphID ($EphID_h^{ctrl}$) with its expiration time ($ExpTime$), and the EphIDs for the MS ($EphID_{ms}$) and the DNS ($EphID_{dns}$) within the AS. The host uses $EphID_h^{ctrl}$ as the source address, and $EphID_{ms}$ and $EphID_{dns}$ as the destination addresses to access the respective services.

Finally, the RS sends the host information (HID, k_{HA}) to infrastructure entities in the AS (e.g., routers, MS, AA); the entities store the information in their database ($host\_info$). The infrastructure of the AS must learn the host information in order to handle packets that are originating from and destined to this host. Specifically, the entities need to learn the HID of the host (HID) and the shared key (k_{HA}) with the host so that they can verify the authenticity of the packets that originate from the host.

4.3 Ephemeral ID Issuance

An EphID is an encrypted token using the AS's secret key (k_A); it contains the host's HID and an expiration time that indicates the validity period for the EphID (i.e., $EphID = E_{k_A}(HID,\ ExpTime)$). Note that the use of encryption enables the issuing AS to obtain the HID and expiration time from an EphID in a stateless fashion, without an additional mapping table.

Every EphID is associated with a public/private key pair (K_{EphID}^+, K_{EphID}^-), which serves three purposes: 1) to mutually authenticate with a peer host, 2) to create a shared key with a peer host for data encryption (Section 4.4.1), and 3) to authenticate shutoff requests (Section 4.5). To keep the data encryption key secret from the AS, the host (not the AS)

[1] In practice, two keys (one for encryption and the other for authentication) are derived from k_{HA}, but for simplicity, we refer to both keys as k_{HA} throughout the protocol description.

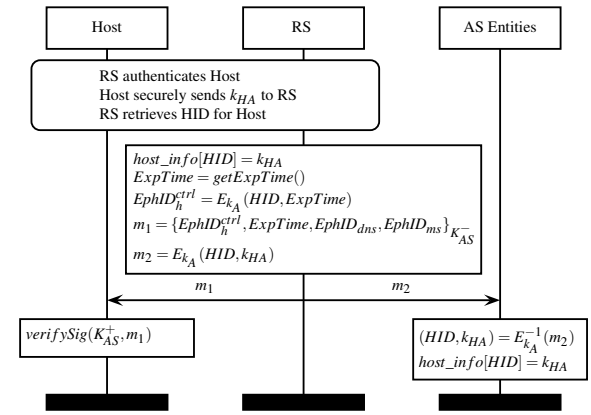


Figure 2: Procedure for Host Bootstrapping.

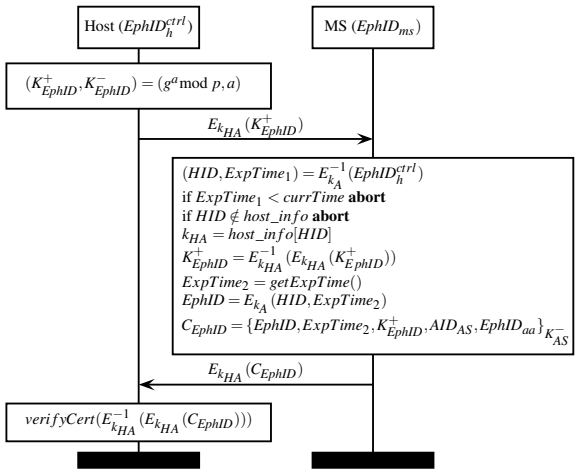


Figure 3: Procedure for EphID Issuance.

generates the key pair and the private key is never revealed to the AS.

The AS certifies the binding between an EphID and a public/private key pair by issuing a short-lived certificate (C_{EphID}) that has the same expiration time as the EphID. From the certificate, a peer host learns the public key (K_{EphID}^+) that is associated to the EphID as well as the EphID expiration time. The certificate additionally contains information about the issuing AS—the AID and the EphID of the accountability agent ($EphID_{aa}$). $EphID_{aa}$ is used by a peer host (with which the requesting host communicates) to initiate the shutoff protocol when necessary (Section 4.5).

To obtain an EphID, the host creates and sends an EphID request message to the MS (Figure 3). Specifically, the host first generates the public/private key pair (K_{EphID}^+, K_{EphID}^-) for the EphID and includes K_{EphID}^+ in the request message. In addition, the host uses $EphID_h^{ctrl}$ as the source address for the request message and encrypts the message using the shared key with the AS (k_{HA}). The message is encrypted to hide it from other entities in the AS that are not part of the AS infrastructure.

If an adversary trying to compromise sender-flow unlinkability (see Section 2.2 for the adversary model) sees the content of EphID request packets, she can identify a common sender across multiple flows at the level of $EphID_h^{ctrl}$. The adversary first learns the ($EphID_h^{ctrl}$, K_{EphID}^+) pair from the EphID request packets; then, the adversary sees the K_{EphID}^+s from the connection establishment packets (see Section 4.4.1), allowing the adversary to identify the common sender of multiple connections. Note that the adversary has not compromised the host identity since only the host's AS can extract host identity from $EphID_h^{ctrl}$. Nonetheless, she has successfully identified a common sender across multiple flows. In APNA, encrypting the EphID request message prevents such attacks.

Upon receiving the request, the MS validates the authenticity of the request; decrypts the source EphID ($EphID_h^{ctrl}$); and performs the following checks: 1) $EphID_h^{ctrl}$ has not expired, 2) the client's identifier (HID) is valid (i.e., has not been revoked), and 3) the message is valid (i.e., the message can be decrypted successfully). If any one of the checks fails, the request is dropped.

Then, the MS proceeds with the EphID issuance: it generates an EphID and creates the short-lived certificate (C_{EphID}) for the EphID. Finally, the MS encrypts the certificate and sends it to the requesting host. The certificate is encrypted for the same reason as encrypting the EphID request packets.

4.4 Data Communication

To communicate, two hosts mutually authenticate each other using each other's certificates and generate a shared symmetric key for their communication session. This key is then used to encrypt all traffic that belongs to this communication session. We emphasize that two hosts can create multiple communication sessions and each session has a different symmetric key to ensure that disclosure of one encryption key does not compromise data privacy of other communication sessions.

4.4.1 Connection Establishment

For every connection establishment between a pair of hosts, the two hosts perform the following tasks: 1) verify each other's EphID certificate that is issued by their corresponding ASes, and 2) establish a shared key via a DH key exchange to encrypt their communication.

Consider two hosts, A and B, with EphIDs $EphID_a$[2] and $EphID_b$, respectively, that are establishing a connection. Assume that the hosts have obtained each other's EphID and the associated certificate (we discuss obtaining EphIDs through DNS in Section 7.1). Using the short-lived certificate of $EphID_b$ and the public-private key pair associated with $EphID_a$, A derives a shared key ($k_{E_aE_b}$) between $EphID_a$ and $EphID_b$. Similarly, B computes the same shared key, completing the

[2]We use small 'a' to denote the EphID issued to A (i.e., $EphID_a$) to emphasize that there can be many EphIDs that are issued to a host.

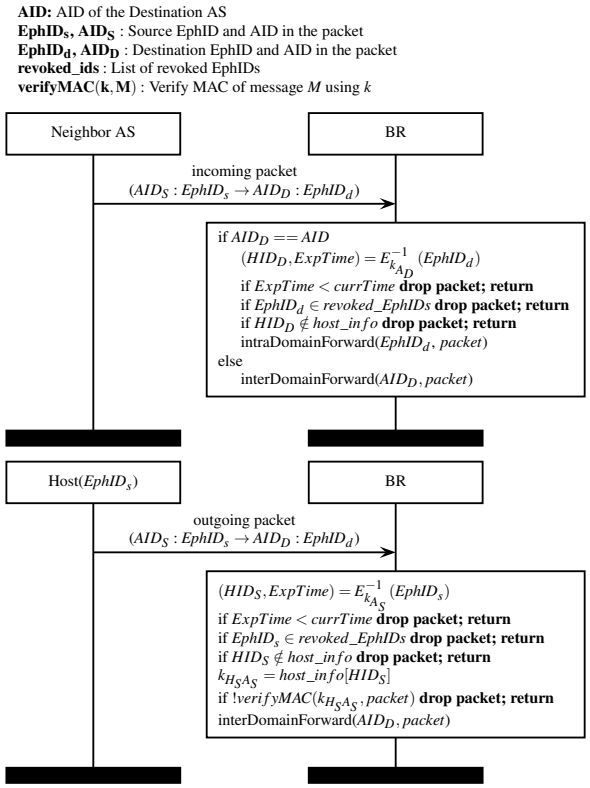


Figure 4: Procedures for Data Packet Forwarding at Border Routers for Incoming (Top) and Outgoing (Bottom) Packets.

connection establishment. This symmetric shared key is then used to encrypt data packets between the two hosts.

4.4.2 Encrypted Communication

After the connection establishment, communication is based on symmetric cryptographic operations. First, the host uses the symmetric key that it shares with the peer to encrypt the packets to the peer. Any existing CCA-secure encryption scheme can be used for the encryption. Second, the host computes a MAC for every packet that it sends, using the symmetric key that it shares with its AS (k_{HA}). This allows the host's AS to link every packet to its source and to drop packets from (potentially) malicious hosts.

4.4.3 Data Forwarding

A border router in the source AS ensures that only packets from authenticated hosts and authorized EphIDs leave the AS; and a border router in the destination AS forwards packets to the correct host based on the destination EphID. Transit ASes do not perform additional operations and simply forward packets to the next AS on the path. As per our design choice, to achieve high-performance data forwarding, only symmetric cryptographic operations are used.

Communication end-points are specified as AID:EphID tuples. For inter-domain forwarding, border routers use only AID to forward packets (Figure 4). Specifically, for external packets entering the AS, a border router checks whether the

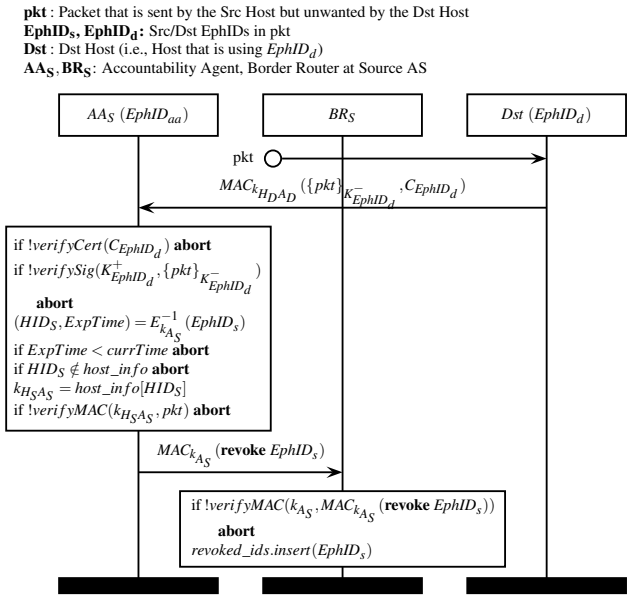


Figure 5: Procedure for Shutoff Protocol.

packet has arrived at the destination AS. If not, the packet is forwarded to the neighboring AS towards the destination AS. At the destination AS, the border router checks the following conditions: 1) the destination EphID ($EphID_d$) is valid (i.e., has not expired and has not been revoked), and 2) HID_D is valid (i.e., is registered and non-revoked).

If all conditions are satisfied, then the packet is forwarded to the destination host: border routers derive the corresponding HID from the EphID and then forward the packet; we assume that intra-domain routers forward packets based on HIDs (e.g., IP addresses).

For outgoing packets, a border router forwards the packets to a neighboring AS only if all of the following conditions are satisfied: 1) the source EphID ($EphID_s$) is valid (i.e., has not expired and has not been revoked), 2) HID_S is valid, and 3) the MAC in the packet is correct.

To verify the MAC in the packet, a border router retrieves the shared key (k_{HA}) between the source host and the AS by searching the host information database ($host\_info$) using the HID of the source host as the key. These checks ensure that only authenticated packets leave the source AS.

4.5 Shutoff Protocol

Shutoff protocols are designed to allow hosts to selectively block traffic from specific source hosts. In our architecture, an accountability agent (AA) checks the validity of a shutoff request and then blocks the source EphID. More specifically, the AA checks whether a customer-host has actually sent the specific packet that the requesting party reports and whether the party is authorized to make the request (e.g., the requesting host was indeed the recipient of the specific packet). The AA checks the validity of the request since, if misused, the shutoff protocol can be used to launch DoS attack against a benign source. Note that the AA does not examine the in-

tent of the source nor tries to determine whether the packet is malicious.

Figure 5 shows the procedure for the shutoff request: the destination host (D) that owns $EphID_d$ is attempting to block traffic coming from the source host (S) that owns $EphID_s$ after receiving a specific packet. The destination host creates a shutoff request message with the following information: 1) the received packet, 2) a signature over the unwanted packet using the private key of $EphID_d$ ($K^-_{EphID_d}$), and 3) the certificate of $EphID_d$. This information serves as evidence that S has indeed sent traffic to D and that the shutoff request is not rogue. Then, D sends the request message to the AA of S.

Upon receiving the request, the AA verifies the certificate of $EphID_d$ and the signature in the request message to confirm that the request has indeed been made by D who owns $EphID_d$. Then, to ensure that the packet has been really generated by S that owns $EphID_s$, the AA checks the authenticity of the packet using the shared key ($k_{H_S A_S}$) with S. Finally, the AA instructs the border routers to revoke $EphID_s$ by putting it into their *revoked_ids* list.

5. IMPLEMENTATION & EVALUATION

We present the implementation and performance evaluation of the architecture's core components—EphID management server and border router.

5.1 EphID Management Server

The EphID Management Server (MS) is responsible for generating EphIDs and for assigning them to hosts. The EphID generation must be efficient since our architecture should even support per-flow EphIDs. We describe the EphID structure, the MS implementation, and then evaluate the performance of the EphID generation procedure.

5.1.1 EphID Structure

We engineer the EphID length to optimize processing for the AES block cipher; it is the only cipher with widespread hardware support, which enables high performance.

An EphID requires the HID of the host and an expiration time (*ExpTime*). AES operates on 16-byte (B) blocks, and we use 4 B for the HID, which are sufficient to uniquely represent all hosts even in large ASes. The expiration time is 2 B long, which is sufficiently large to express one day in two-second granularity.

Recall that the security requirement for EphIDs is a CCA-secure encryption scheme. To this end, we use a generic composition called Encrypt-then-MAC [21] that combines a symmetric encryption with a message authentication code (MAC) (Figure 6). The concatenation of *HID* and *ExpTime* is first encrypted using AES in counter mode. Secure operation of this mode requires a unique initialization vector (IV) for every encryption (i.e., for every EphID). Moreover, the use of the IV allows us to generate multiple EphIDs for a single *HID*. Note that the plaintext data is shorter than a single

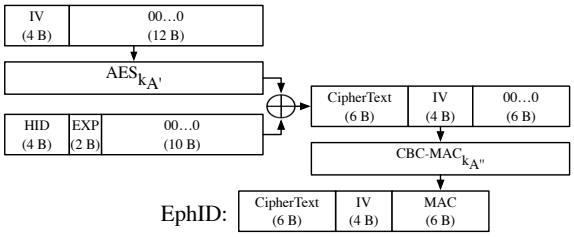


Figure 6: EphID Construction, where ExpTime is EXP.

AES block (16 B) and thus the input must be padded to 16 B; the one-block plaintext requires a single AES operation.

Next, a MAC is computed. The MAC is computed over the first 6 B of the previously generated ciphertext and the IV that was used in that encryption. We use CBC-MAC based on AES to generate the MAC, which is secure as we have a fixed input length.

Finally, the EphID is constructed from the 6 B of the ciphertext, 4 B IV, and 6 B of the MAC; the total length is 16 B. Note that the keys used for encryption ($k_{A'}$) and authentication ($k_{A''}$) are different; however, they are derived from the secret key of the AS (k_A).

5.1.2 MS Implementation

The MS generates EphIDs according to the procedure in Figure 3. For asymmetric cryptography, we use cryptographic primitives based on Curve25519 [22], which offers high performance and features small public-keys (32 B) and small signatures (64 B). Key exchange is done using the elliptic-curve variant of Diffie-Hellman (ECDH). To create digital signatures for certificates, we use the Ed25519 signature scheme [23] and the Ed25519 SUPERCOP REF10 implementation.[3] For symmetric cryptographic operations, we leverage Intel's AES-NI encryption instruction set [24]. Furthermore, we implement the host database (*host_info*) that stores the shared keys between hosts and the AS as a hash table using HID as the key.

As an optimization, we parallelize the EphID generation by using four processes to simultaneously handle EphID requests. The parallelization is straightforward since the generation does not require any coordination between the processes (e.g., shared memory or inter-process communication). However, no other optimizations were performed (e.g., optimizing the Ed25519 REF10 implementation).

5.1.3 MS Performance Evaluation

We demonstrate the efficiency of generating per-flow EphIDs. To this end, we need statistics for the peak flow generation rate inside an AS.

We use a 24-hour packet trace of HTTP(S) traffic from SWITCH.[4] The trace contains over 104 million entries for HTTP traffic and 74 million entries for HTTPS traffic. Each entry contains a timestamp and anonymized source/destination

[3] http://bench.cr.yp.to/supercop.html

[4] The Swiss academic ISP (www.switch.ch).

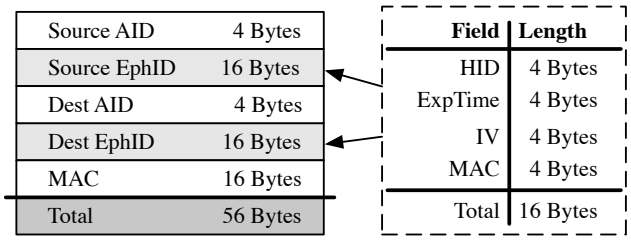
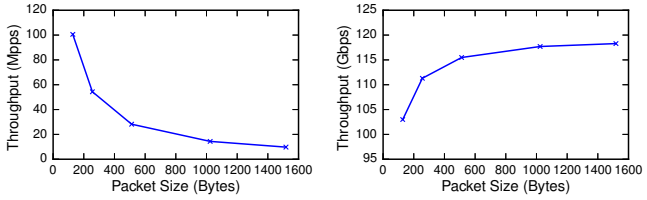


Figure 7: APNA Header Information and EphID Field Lengths.



(a) (b)

Figure 8: Forwarding performance expressed as (a) packet-rate and (b) bit-rate.

IDs. We identify 1,266,598 unique hosts generating a peak rate of 3,888 active HTTP(S) sessions per second.

We test our implementation on a desktop machine with an Intel Core i5-3470s CPU (4 cores, 2.9GHz) and 4 GB of DDR3 memory. For 500,000 EphID requests, our implementation runs for 6.87 seconds. On average, 13.7 µs are needed for a single EphID generation, translating to a generation rate of 72.8k EphIDs/sec—over 18 times higher than the request rate. Our experiment shows that even a low-end desktop machine can keep up with the traffic demands of a real AS that has over 1.2 million hosts.

5.2 Border Router

We describe our border router prototype starting with the structure of the network header. Then, we describe the border router implementation and evaluate the forwarding performance.

5.2.1 APNA Header Information

The network header information (Figure 7) contains the source and destination end points (expressed as AID:EphID tuples) and a MAC over the packet's content. We use 4 B to express the AID since 4 B are used for AS numbers in the Internet; the EphID field requires 16 B as described in Section 5.1.1; the MAC field requires 16 B. The fields in the packet header sum up to 56 B.

5.2.2 Border Router Implementation

Our border router performs additional processing compared to traditional IPv4/IPv6 forwarding (Figure 4). Namely, the border router additionally performs one decryption, two table lookups, and one MAC verification. For MAC computation, we use Cipher-based Message Authentication Code (CMAC) that is secure for variable-length inputs [25].

We use DPDK [26] as our packet processing platform, which allows us to implement the required functionality in userspace. The decryption of the EphID in the packet is implemented through Intel AES-NI [24].

5.2.3 Forwarding Performance Evaluation

We evaluate the forwarding performance on a commodity server with two Intel Xeon E5-2680 CPUs and two non-uniform memory access (NUMA) nodes; each NUMA node has four banks of 8 GB DDR3 RAM. The server is equipped with 6 dual-port 10 GbE NICs, providing a total capacity of 120 Gbps. To generate traffic, we use Spirent-SPT-N4U-220 [27] connected back-to-back with the server. The server receives the traffic, processes it, and sends it back to the generator.

We perform a throughput experiment for 5 different packet sizes—128, 256, 512, 1024, and 1518-byte packets. The results (Figure 8) confirm that we are able to perform the required additional processing without incurring a throughput penalty. The measured performance matches the theoretical maximum performance; we omit this performance line for demonstration purposes since the two lines match. Figure 8(a) shows that even for small packet sizes (i.e., high packet-rates), the border router performs optimally. Figure 8(b) shows that as packet sizes increase, we saturate the capacity of 120 Gbps. The border router performs optimally because the additional operations are lightweight. The border router's CPUs have adequate capacity to perform this processing without degrading performance for the given packet rates. Under higher packet rates, the heavier load would start to degrade forwarding performance slightly.

6. SECURITY ANALYSIS

We demonstrate how APNA prevents attacks that undermine source accountability and data privacy.

6.1 Attacking Source Accountability

An adversary attacking source accountability (Section 2.1) has three attack vectors at hand.

EphID Spoofing: The adversary can attempt to use an EphID that is issued to another host (the spoofed victim). For instance, an adversary that shares the same access port with the victim can sniff traffic and observe valid EphIDs that are in use. However, using such an EphID is not sufficient since every outgoing packet has to contain a MAC that is computed with the shared key between the host and the host's AS. Without the corresponding shared key, the adversary cannot create valid MACs, resulting in spoofed packets that are dropped by the host's AS (additionally making the attack visible). Obtaining the shared key requires compromising the host or the AS; our adversary model does not account for such compromises.

An active adversary can attempt to obtain an EphID by pretending to be another host. However, such an attack is

infeasible: the adversary not only needs to learn the control EphID ($EphID_h^{ctrl}$) of the victim, but also needs to learn the shared key between the victim and the source AS.

Unauthorized EphID Generation: The adversary can attempt to create an unauthorized EphID. However, such an attempt is not feasible since the EphID construction (Figure 6) is CCA-secure.

We achieve a CCA-secure encryption scheme through two primitives. First, we use symmetric encryption in counter mode with a fresh IV for every encryption; this encryption is secure under a chosen plaintext attack. Second, we use a CBC-MAC scheme to authenticate the concatenation of the ciphertext and the IV. Note that our use of the CBC-MAC is secure since the input length to the CBC-MAC is fixed to 16 B.[5] The combination of these two primitives results in a CCA-secure encryption scheme [21].

Identity Minting: A common attack against systems that provide accountability is identity minting, whereby a malicious host attempts to create multiple (unauthorized) identities. In APNA, a host can at most have one HID at any moment; and, a new HID invalidates previous HID and all associated EphIDs.

Unauthorized Shutoff Requests: The shutoff protocol can be misused to perform a DoS attack against a host. To prevent unauthorized shutoff requests, three measures are implemented. First, only the destination host and destination AS are authorized to issue a shutoff request. Furthermore, the shutoff requester has to present the unwanted packet, which proves that the source has indeed sent the packet. Since every packet has been cryptographically marked by the source AS, the destination cannot issue a shutoff request with a rogue packet. Lastly, the shutoff requester must present its authorization credentials—it needs to sign the request message with the private key associated with the destination EphID, and include the corresponding short-lived certificate in the request message, proving that it is an authorized party.

6.2 Attacking Privacy

An adversary attacking data privacy can attempt to eavesdrop on communication data or store it and decrypt it once she obtains the encryption keys. In APNA, traffic is encrypted by default and our scheme achieves perfect forward secrecy: The symmetric key that is used for data encryption is bound to the EphID (and the public/private key pair for that EphID) that is used for the corresponding communication session. This key pair is not used to derive other encryption keys and is not derived from other long-term private keys (K_{AS}^{-}). Hence, only the compromise of a private key for an EphID compromises data privacy and only for the communication session that uses this EphID.

Alternatively, an AS-level adversary can actively try to compromise data privacy of a customer host through a MitM attack. The malicious AS can perform a MitM attack during the shared key establishment between the victim ($EphID_v$) and a peer host ($EphID_p$). In this attack the malicious AS replaces the certificate for the EphID of the victim host (C_{EphID_v}) with another (fake) certificate, pretending to be the victim host to the peer host; the peer host accepts C_{EphID_v}. However, the AS cannot deceive the victim by pretending to be the peer host because it cannot generate the certificate for $EphID_p$ (C_{EphID_p}) that is signed by the private key of the peer host's AS. Consequently, the connection is not established and the adversary cannot read any communication of the hosts. The MitM attack is only possible if the source and destination ASes collude, which we do not consider.

For communication between two hosts in the same AS (i.e., intra-domain communication), APNA does not provide any privacy guarantee from the AS: the identities of the two hosts are already known to the AS (compromising host privacy), and the AS can perform MitM attacks to decrypt communication between the hosts (compromising data privacy) as the AS can fake both certificates for the EphIDs that the hosts use. The two hosts can use security protocols at higher layers (e.g., TLS) to encrypt their communication or use onion routing, such as Tor [5], to make a detour.

7. PRACTICAL CONSIDERATIONS

7.1 DNS Registration

Today, the names of publicly accessible services (e.g., an online shopping website) are typically registered to public DNS servers. In APNA, the servers that host public services publish the EphID to a DNS server, and the DNS server returns the EphID with the corresponding certificate for a requested domain name. To this end, the server performs two tasks: 1) it requests an EphID and the associated certificate from its AS; and 2) it registers the certificate under the domain name to DNS;[6] the registered EphID will be used as the destination address in future communication.

Publishing certificates to the DNS raises a problem: a shutoff request against a published EphID would terminate any ongoing communication sessions that use this EphID. A naïve solution is to update the DNS entry with a new EphID whenever the published EphID becomes invalid. However, this would become burdensome for the DNS infrastructure, if attackers continuously issue shutoff requests against a domain.

Our solution is to define *receive-only EphIDs*—EphIDs that are used only to receive packets and are never used as the source EphIDs. Since they are never used as the source identifier, they cannot become the target of shutoff requests. To avoid using receive-only EphIDs as the source identifier, the communication establishment to a server needs to be changed (i.e., the server does not respond to the client using the receive-only EphID).

Client-Server Connection Establishment: To support receive-only EphIDs by the server, the connection establishment procedure in Section 4.4.1 is extended. To simplify the narrative, assume that the client uses $EphID_c$ to connect to the server, and that the server uses $EphID_r$ as the receive-only EphID and $EphID_s$ to serve the client.

After obtaining $EphID_r$ from DNS, the client contacts the

[5] CBC-MAC is insecure for variable-length messages [25].

[6] We assume DNSSec to authenticate DNS records.

server using $EphID_c$ and $EphID_r$ as the source and destination EphIDs, respectively. The server verifies the short-lived certificate of $EphID_c$ and computes a shared key that will be used to encrypt data packets between the client and the server. However, instead of using the short-lived certificate of $EphID_r$, the server uses the short-lived certificate of $EphID_s$ to compute the shared key. Then, in the response message to the client, the server includes the short-lived certificate of $EphID_s$ to inform the client that $EphID_s$ will be used by the server to serve the client.

The client verifies the short-lived certificate of $EphID_s$ and computes the shared key using the certificates for $EphID_s$ and $EphID_c$. Then, the client uses $EphID_s$ as the destination EphID to communicate to the server.

Protecting DNS Queries: If the DNS server is operated by the host's AS, the AS can compromise the privacy of the DNS query—the AS knows the identity of the host from the EphID and retrieves the content of the query from the DNS server. To prevent such a compromise, the host can use a DNS server that he trusts and that is not operated by the AS that he resides in.

7.2 Deployment in today's Internet

Our ideas are not restricted to a certain Internet architecture, thus APNA could be used in today's Internet: IPv4 addresses of hosts serve as HIDs; IPv4 addresses of APNA routers in ASes serve as AIDs; Generic Routing Encapsulation (GRE) [28] is used to insert the APNA header into IPv4 packets; and, APNA gateways mediate between IPv4 and the APNA protocol, leaving the host's operating system unchanged. The following paragraphs summarize the deployment strategy in the Internet; our technical report [29] contains more detailed descriptions.

For intra-domain forwarding in the source AS, the source host puts its IP address and the IP address of an APNA router in the source AS as the source and destination addresses in the IPv4 header (that comes before the GRE header), respectively. For intra-domain forwarding in the destination AS, an APNA router 1) decrypts the destination EphID in the APNA header to get the HID (i.e., IPv4 address) of the destination host; and, 2) replaces the destination IPv4 address of the IPv4 header with the HID.

This intra-domain forwarding has a privacy implication. Within the source and destination ASes, the addresses of the hosts are visible; hence, it is not possible to provide any privacy guarantee against an adversary who observes packets within the ASes. However, once an AS fully deploys APNA (i.e., all routers forward packets based on EphIDs), this privacy leakage can be mitigated.

For inter-domain forwarding in the source AS, an APNA router replaces the addresses in the IPv4 header of the APNA packet with its IPv4 address and the destination AID as the new source and destination addresses, respectively. For all transit ASes, the packet is forwarded based on the destination address in the IPv4 header.

We leverage the GRE protocol to interconnect two APNA entities (e.g., border routers) over the IPv4 network. This enables encapsulation of the APNA header after the GRE tunnel header; a protocol number assigned by IANA and used for the *Protocol Type* field in the GRE header would indicate that the encapsulated protocol is APNA.

Furthermore, gateways are used to bridge between today's Internet and APNA, without having to update the host network stack. The role of the gateway is two-fold: 1) as an APNA host, it runs the protocols described in Section 4; and 2) as a packet translator, it converts between native IPv4 and APNA packets.

Hosts Behind NAT Devices. In order to support NATs, the NAT acts as a small AS for its hosts, while it acts as a single host for the AS network. Thus, the NAT acts as a RS, a MS, a router, and an accountability agent on behalf of its clients. As a RS, the NAT bootstraps the hosts into the AP's internal network. As an MS, the NAT makes EphID requests on behalf of its hosts to the AS. As a router, the NAT implements the data forwarding procedures described in Figure 4. The NAT plays these roles following the same protocols as described for an AS, only with minor differences [29].

As an accountability agent, the NAT identifies the misbehaving hosts based on EphIDs. Since the hosts behind a NAT are not visible to the AS and since the AS issues EphIDs to the NAT not to the hosts, the AS holds the NAT accountable for misbehaving EphIDs. Then, the NAT determines the host that is using the misbehaving EphID.

8. DISCUSSION

8.1 Ephemeral ID Granularity

Thus far, we have argued that APNA does not impose the granularity at which EphIDs should be used and we have shown that the EphIDs can be generated at high speed (see Section 5.1.3). In this section, we present four granularities at which EphIDs can be used.

Per-Flow Ephemeral ID: We expect this to be the typical use case where a host uses different EphIDs for different flows. There are two advantages to per-flow EphIDs. First, it prevents an observer's attempt to identify a common sender of multiple flows by inspecting the content of the packets (i.e., APNA header and payload). Second, shutoff incidents have limited impact on a host. It terminates the flow that uses the reported EphID as the source; however, all other flows remain intact. The disadvantage of this case is that a host needs to acquire and manage EphIDs for every new flow.

Per-Host Ephemeral ID: A host uses a single EphID for all packets. The advantage of this model is that a host only needs to acquire and manage one EphID. However, there are two drawbacks. Since all packets have the same source EphID, all packets are linked to a common sender; and, a shutoff request terminates all connections from the host.

Per-Packet Ephemeral ID: A host could use a different EphID for each packet so that it is difficult to link packets to a single flow. This provides the strongest privacy guarantee; however, additional mechanisms are necessary for the destination host to demultiplex packets into flows [30].

Per-Application Ephemeral ID: An EphID can be used to represent all packets that are generated by an application or a service that is running on the host. This EphID granularity facilitates managing traffic that is generated by an application. For example, if an AS enforces its hosts to use per-application EphIDs, the AS and its hosts could collaboratively identify malicious applications (e.g., a bot) running at the hosts. The network identifies malicious activities (e.g., flooding attacks) to a source EphID and inform the host about the EphID; then, the host identifies the application that uses the EphID and takes appropriate actions.

8.2 Support for Network Feedback

In APNA, an on-path intermediate node (e.g., router) can send messages (e.g., ICMP) back to the source host.

Assume that router R is attempting to send an ICMP message to the source host S. To send an ICMP message to S, R uses the source EphID (i.e., $EphID_s$) from the received packet (that has prompted the ICMP message), and uses one of its EphIDs (i.e., $EphID_r$) as the destination and source EphIDs, respectively. Hence, APNA protects the identity of R from the source host while holding R accountable for the ICMP message.

8.3 Parameter Considerations

Expiration Time for EphIDs: If EphIDs are used per flow, the expiration time can be set to 15 minutes as 98% of the flows in the Internet last less than 15 minutes [31]. Alternatively, the EphID issuance protocol (Section 4.3) can be extended to allow hosts to express their choice of expiration time. For instance, an AS may specify three categories (short-term, medium-term, long-term EphIDs) to accommodate diverse flow duration times.

Managing Revoked EphIDs: EphIDs can be preemptively revoked before they expire: a host could revoke an EphID that is no longer needed, or an EphID could have been subjected to a shutoff incident. Regardless of the reason for revoking EphIDs, border routers in the ASes need to store a list of revoked EphIDs (i.e., *revoked_EphIDs* in Figure 4). If there are too many revocations in an AS, it burdens the border routers since the size of the *revoked_EphIDs* would become large.

There are methods to limit the size of the *revoked_EphIDs* list. First, an AS can rotate its secret key (k_A) periodically; the rotation invalidates all previously issued EphIDs. Second, since EphIDs will expire over time and packets using expired EphIDs are dropped, the expired EphIDs can be removed from *revoked_EphIDs*. Third, if too many EphIDs of a host are revoked, the AS should view it as a sign of malicious activity by the host. In such an event, the AS revokes the HID of the host invalidating all EphIDs that are issued to the host and it assigns a new HID to the host. In addition, the AS can contact the host for corrective measures. Such approaches are already in use, e.g., ISPs that participate in the Copyright Alert System (CAS) [32].

8.4 Handling Replay Attacks

A malicious entity that aims to "harm" a source host may replay packets of the source. In the short-term, replayed packets may induce shutoff incidents against the source host, disrupting communication of the source; and in the long-term, the AS of the source host may take retributive action against the source host for repeated shutoff incidents.

Replay attacks can be prevented by making every packet unique. That is, a nonce field is added to the APNA header (Figure 7), and a source host puts a unique number for each generated packet. Then, the destination host performs replay detection based on the nonces in the packets and discards all duplicate packets.

Ideally, replayed packets should be filtered near the replay location, but this requires routers in the network to perform replay detection. Designing a practical in-network replay detection mechanism that does not affect routers' forwarding performance is not trivial; it is our future work to design such a mechanism.

8.5 Governments and Privacy

Although generally perceived as a threat on communication privacy, there are legitimate reasons for governments to subvert communication privacy (e.g., to monitor terrorist activities). In fact, many governments by law mandate ISPs to keep a record of their traffic (e.g., source and destination IP addresses, packet content, etc).

APNA protects communication privacy by making mass surveillance difficult; however, at the same time, it allows entities, such as a government, to deanonymize communication when necessary. With the cooperation of an AS, a government can deanonymize the identity of hosts from EphIDs. Furthermore, if the government has cooperation from the ASes in which communicating hosts reside, the AS could decrypt ongoing communication by performing a MitM attack. However, the government cannot observe communication by simply collecting packets since packets are encrypted, which makes mass surveillance difficult. In addition, since APNA achieves perfect forward secrecy, governments cannot decrypt past communication sessions of a host even if the long-term public key of the host is disclosed.

9. RELATED WORK

Persona [33] seeks to balance privacy and accountability at the network layer. The source ISP replaces the IP address of each outgoing packet with another address from an assigned pool. This approach hides the source's identity, but it breaks the notion of flow and prevents the destination from demultiplexing connections.

Accountable and Private Internet Protocol (APIP) [6] proposes an architecture that balances accountability and privacy at the network layer. In APIP, the source address in the network header is replaced with the address of an *accountability delegate* that vouches for the source's packets. The return address can then be specified at a higher layer—invisible from the network—protecting the source's privacy.

Senders are expected to brief each packet to their accountability delegate such that on-path devices can request a "vouching proof" from the corresponding delegate.

APIP balances privacy and accountability at the network layer, but it comes with certain limitations. APIP's notion of privacy is limited to sender-flow unlinkability, leaving data privacy and the associated challenges (e.g., key distribution, management, and establishment) unaddressed. Our proposal presents a holistic architecture that addresses these drawbacks and by default supports data privacy. Furthermore, the design of APIP precludes every packet from being accounted for in the network: it is possible for a malicious host to omit reporting packets to its accountability delegate when the flow for those packets has been "whitelisted".[7] In APNA, every packet is linked to its sender since a MAC is computed using the shared key between the AS and the host for every packet (Section 4.4). Second, masking the return address complicates getting messages from the network back to the source—the messages must be redirected through the accountability delegate of the source; the complexity of this functionality remains unaddressed. APNA allows the network to send messages directly to the source while preserving host privacy and the accountability properties (Section 8.2).

Source Accountability: The Accountable Internet Protocol (AIP) [11] treats source accountability as a central principle. In AIP, self-certifying IDs and a shutoff protocol (implemented by smart Network Interface Cards) are used to identify and block malicious sources. Our architecture uses self-certifying IDs in an anonymity-preserving way and delegates the shutoff functionality to the source domain.

In Passport [34], OPT [35] and ICING [36], MACs are used for each AS on the end-to-end path, allowing on-path ASes to verify the authenticity of packets. In APNA, such mechanisms can be used to strengthen the shutoff mechanism [29].

Bender et al. [37] were first to introduce the concept of accountability agents in their Accountability-as-a-Service (AaaS) proposal. However, AaaS does not address privacy considerations and requires symmetric keys between all AS pairs.

Host Privacy and Anonymity: Raghavan et al. [38] propose ISP-wide NATs to hide the hosts' identities from entities in other ASes. We borrow their motivation that the size of today's large ISPs provides sufficiently large anonymity sets. Onion routing [4] and Mix Networks [39] by design provide anonymity at the cost of source accountability.

Han et al. [40] propose a cross-layer design that uses pseudonyms to hide the user's identity. Similar to APNA, the proposal allows the user to choose the level of anonymity and it uses encryption to mask the identity of the host in network addresses. However, the proposal falls short of being a complete architecture that balances between accountability and privacy: it does not consider pervasive data encryption and the associated challenges, such as certificate management, and does not consider source accountability.

Data Privacy: Farrell and Tschofenig [1] argue that pervasive monitoring—defined as the widespread and often covert surveillance through intrusive gathering of communication information—is a widespread attack on privacy. In response, Kent [41] proposes pervasive encryption as a countermeasure against pervasive monitoring. In a related effort, the Let's Encrypt[8] organization encourages the use of encrypted web traffic by issuing free TLS certificates for web servers. Our proposal does not replace transport-layer encryption, but rather promotes pervasive encryption to a fundamental design tenet of the network layer. In addition, we propose a concrete solution for key distribution, establishment, and management.

MinimaLT [42] proposes an architecture that supports pervasive data encryption and achieves PFS at low latency; however, MinimalLT does not consider source accountability. In Section 4.4.1, we show how our architecture supports data privacy with PFS while enforcing source accountability.

10. CONCLUSIONS

We propose APNA, an architecture that resolves the accountability-privacy tussle by enlisting ISPs as accountability agents and privacy brokers. As accountability agents, ISPs authenticate hosts and their packets; and as privacy brokers, ISPs anonymize the identities of communicating parties and assist in the establishment of shared keys for end-to-end data encryption. Even with additional tasks that an AS has to perform, we have demonstrated that our architecture can be implemented on commodity hardware and scale to large networks without adding substantial overhead.

By facilitating (and by enabling by default) pervasive encryption between endpoints, APNA can help frustrate adversaries conducting indiscriminate mass surveillance. At the same time, APNA can assist in lawful, targeted requests for subscriber communications, since ISPs can comply with data retention laws by storing customer to EphID bindings as well as the packets. However, abuse of such requests for information are minimized due to the perfect forward secrecy of our scheme.

11. ACKNOWLEDGMENTS

We would like to thank Yue-Hsun Lin for the initial discussion; and the anonymous reviewers and our shepherd, Roya Ensafi, for their insightful feedback and suggestions. The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013)/ERC grant agreement 617605; and from the Institute for Information & communications Technology Promotion (IITP) grant funded by the Korean government (MSIP)/No.B0717-16-0040. We also gratefully acknowledge support by ETH Zürich and by Intel for their equipment donation that enabled the high-capacity experiments.

[7] Verifiers do not verify flows that have been "whitelisted," and a sender does not brief packets unless it is asked by its accountability delegate under the recursive verification method (Section 5 in APIP [6]).

[8] https://letsencrypt.org

12. REFERENCES

[1] S. Farrell and H. Tschofenig, "Pervasive Monitoring Is an Attack," RFC 7258, IETF, 2014.

[2] "NSA Spying," https://www.eff.org/nsa-spying.

[3] K. McCarthy, "World's Largest Internet Exchange Sues Germany over Mass Surveillance," http://www.theregister.co.uk/2016/09/16/ixp_sues_german_govt_surveillance/.

[4] M. G. Reed, P. F. Syverson, and D. M. Goldschlag, "Anonymous Connections and Onion Routing," *Selected Areas in Communications, IEEE J. on*, 1998.

[5] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The Second-generation Onion Router," in *Proc. of USENIX Security Symposium*, 2004.

[6] D. Naylor, M. K. Mukerjee, and P. Steenkiste, "Balancing Accountability and Privacy in the Network," in *Proc. of ACM Conference on SIGCOMM*, 2014.

[7] S. Kent and K. Seo, "Security Architecture for the Internet Protocol," RFC 4301, IETF, 2005.

[8] C. Kaufman, "Internet Key Exchange (IKEv2) Protocol," RFC 4306, IETF, 2005.

[9] B. Rowe, D. Wood, D. Reeves, and F. Braun, "The Role of Internet Service Providers in Cyber Security," http://goo.gl/t7f7NA, 2011.

[10] D. G. Lichtman, "Holding Internet Service Providers Accountable," *University of Chicago Coase-Sandor Working Paper Series in Law and Economics*, 2004.

[11] D. G. Andersen, H. Balakrishnan, N. Feamster, T. Koponen, D. Moon, and S. Shenker, "Accountable Internet Protocol (AIP)," in *Proc. of ACM Conference on SIGCOMM*, 2008.

[12] A. Pfitzmann and M. Hansen, "Anonymity, Unobservability, and Pseudonymity: A Consolidated Proposal for Terminology," http://goo.gl/GExcmQ, Jul 2000.

[13] A. J. Menezes, P. C. v. Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, fifth printing ed. CRC Press, 2001.

[14] V. Shmatikov and M.-H. Wang, "Timing Analysis in Low-latency Mix Networks: Attacks and Defenses," in *Proc. of European Conference on Research in Computer Security (ESORICS)*, 2006.

[15] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, "Peek-a-Boo, I Still See You: Why Efficient Traffic Analysis Countermeasures Fail," in *Proc. of IEEE Symposium on Security and Privacy (S&P)*, 2012.

[16] M. Juarez, M. Imani, M. Perry, C. Diaz, and M. Wright, "Toward an Efficient Website Fingerprinting Defense," in *Proc. of European Conference on Research in Computer Security (ESORICS)*, 2016.

[17] J. Hayes and G. Danezis, "k-fingerprinting: A Robust Scalable Website Fingerprinting Technique," in *Proc. of USENIX Security Symposium*, 2016.

[18] ARIN, "Resource Public Key Infrastructure," http://bit.ly/1EJCQoT.

[19] C. Rigney, S. Willens, A. Rubens, and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)," RFC 2865, IETF, 2000.

[20] V. Fajardo, J. Arkko, J. Loughney, and G. Zorn, "Diameter Base Protocol," RFC 6733, IETF, 2012.

[21] M. Bellare and C. Namprempre, "Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm," in *Advances in Cryptology—ASIACRYPT*, 2000.

[22] D. J. Bernstein, "Curve25519: New Diffie-Hellman Speed Records," in *Public Key Cryptography (PKC)*, 2006.

[23] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang, "High-speed High-security Signatures," *J. of Cryptographic Engineering*, 2012.

[24] S. Gueron, "Intel Advanced Encryption Standard (AES) New Instruction Set," Mar 2010.

[25] M. Bellare, J. Kilian, and P. Rogaway, "The Security of the Cipher Block Chaining Message Authentication Code," *J. of Computer and System Science*, 2001.

[26] "Data Plane Development Kit," http://dpdk.org.

[27] "Spirent SPT-N4U-220 Chassis," http://www.spirent.com/Ethernet_Testing/Platforms/N4U_Chassis.

[28] D. Farinacci, T. Li, S. Hanks, D. Meyer, and P. Traina, "Generic Routing Encapsulation (GRE)," RFC 2784, IETF, 2000.

[29] T. Lee, C. Pappas, D. Barrera, P. Szalachowski, and A. Perrig, "Source Accountability with Domain-brokered Privacy," https://arxiv.org/abs/1610.00461, 2016.

[30] T. Lee, C. Pappas, P. Szalachowski, and A. Perrig, "Communication Based on Per-Packet One-Time Address," in *Proc. of IEEE Conference on Network Protocols (ICNP)*, 2016.

[31] N. Brownlee and K. Claffy, "Understanding Internet Traffic Streams: Dragonflies and Tortoises," *IEEE Communication Magazine*, 2002.

[32] "The Copyright Alert System," http://goo.gl/I9GsOl.

[33] Y. Mallios, S. Modi, A. Agarwala, and C. Johns, "Persona: Network Layer Anonymity and Accountability for Next Generation Internet," *IFIP Advances in Information and Communication Technology*, 2009.

[34] X. Liu, A. Li, X. Yang, and D. Wetherall, "Passport: Secure and Adoptable Source Authentication," in *Proc. of USENIX Conference on Networked Systems Design and Implementation (NSDI)*, 2008.

[35] T. H.-J. Kim, C. Basescu, L. Jia, S. B. Lee, Y.-C. Hu, and A. Perrig, "Lightweight Source Authentication and Path Validation," in *Proc. of ACM Conference on SIGCOMM*, 2014.

[36] J. Naous, M. Walfish, A. Nicolosi, D. Mazières, M. Miller, and A. Seehra, "Verifying and Enforcing Network Paths with ICING," in *Proc. of ACM Conference on Emerging Networking EXperiments and Technologies (CoNEXT)*, 2011.

[37] A. Bender, N. Spring, D. Levin, and B. Bhattacharjee, "Accountability as a Service," in *Proc. of USENIX Worskhop on Steps to Reducing Unwanted Traffic (SRUTI)*, 2007.

[38] B. Raghavan, T. Kohno, A. C. Snoeren, and W. David, "Enlisting ISPs to Improve Online Privacy: IP Address Mixing by Default," in *Proc. of Privacy Enhancing Technologies Symposium (PETS)*, 2009.

[39] D. L. Chaum, "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms," *Communications of the ACM*, 1981.

[40] S. Han, V. Liu, Q. Pu, S. Peter, T. Anderson, A. Krishnamurthy, and D. Wetherall, "Expressive Privacy Control with Pseudonyms," in *Proc. of ACM Conference on SIGCOMM*, 2013.

[41] S. Kent, "Pervasive Encryption as a Countermeasure to Pervasive Monitoring," draft-kent-pervasive-encryption-00, IETF, 2014.

[42] M. W. Petullo, X. Zhang, J. A. Solworth, D. J. Bernstein, and T. Lange, "MinimaLT: Minimal-latency Networking Through Better Security," in *Proc. of ACM Conference on Computer & Communications Security (CCS)*, 2013.