Diss. ETH No. 24720

Towards an Accountable and Private Internet

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES of ETH ZURICH
(Dr. sc. ETH Zurich)

presented by

TAE-HO LEE

Master of Science in Electrical Engineering,
Stanford University

born on 24.02.1986

citizen of South Korea

accepted on the recommendation of

Prof. Dr. Adrian Perrig
Prof. Dr. Heejo Lee, co-examiner
Prof. Dr. Timothy Roscoe, co-examiner
Prof. Dr. Peter Steenkiste, co-examiner

2017

Abstract

Today's Internet fails to provide either source accountability or privacy; packets on the Internet cannot be attributed to their source, and communications on the Internet cannot be kept private. Furthermore, even if the Internet can be modified to support the two properties, simultaneously providing both is difficult because they are conflicting. Hence, most future Internet architecture (FIA) proposals have focused on one property at the expense of the other.

This dissertation explores an FIA that balances between source accountability and privacy. To this end, we investigate a trust model that can balance the two properties. Then, based on the trust model, we design a data plane that balances the properties and satisfies today's packet forwarding requirements. Lastly, we explore how to deploy such FIAs.

Logically, this dissertation is organized into two parts: the first part explores the design of an FIA that balances between source accountability and privacy, and the second part explores deployment strategies for such an FIA. The first part makes the following three contributions. First, we propose the *Accountable and Private Network Architecture (APNA)* that guarantees source accountability while preserving the privacy of the users. Second, we strengthen the privacy guarantees of APNA. To this end, we define a new concept, *flow-packet unlinkability*—preventing an adversary from linking packets to flows. Then, we extend APNA to achieve flow-packet unlinkability. Lastly, we improve the source accountability guarantees by eliminating vulnerabilities introduced by replayed packets. Specifically, we first propose an attack that can be launched by simply replaying packets using compromised routers. Then, we describe an in-network packet replay suppression mechanism to mitigate such attacks.

The second part of the dissertation explores two different approaches to deploy FIAs—*integrated* and *overlay* approaches. The overlay approach deploys the FIA itself by constructing an overlay over today's Internet. In contrast, the integrated approach deploys the FIA by integrating its main ideas into today's Internet; the FIA itself does not need to be deployed. This approach uses the FIA research as a means to explore new and innovative ideas without confining oneself to the rigidity of today's Internet. Then, these new ideas are realized on today's Internet using well-established

technologies. As an example, we take the key ideas of APNA and realize them on today's Internet to enhance privacy.

Zusammenfassung

Das heutige Internet unterstützt weder Rechenschaftspflicht noch Privatsphäre von Sendern. Über das Internet versandet Pakete können nicht mit Bestimmtheit ihren Sendern zugeordnet werden und die Vertraulichkeit von Kommunikationen über das Internet kann nicht gewährleistet werden. Selbst wenn man Änderungen an der Architektur des Internets vornehmen könnte, macht die Gegensätzlichkeit dieser Eigenschaften es schwierig beide gleichzeitig zu unterstützen. Aus diesem Grund richten die meisten Vorschläge einer "Future Internet" Architektur (FIA) ihre Aufmerksamkeit auf eine dieser beiden Eigenschaften.

Diese Dissertation untersucht eine FIA, die zwischen Rechenschaftspflicht und Privatsphäre von Sendern abwägt. Zu diesem Zweck erforschen wir ein Vertrauensmodell, das ein Gleichgewicht zwischen diesen sich widersprechenden Eigenschaften schafft. Basierend auf diesem Modell entwerfen wir eine Datenbeförderungsinfrastruktur, die diese Eigenschaften erreicht und die Anforderungen heutiger Paketweiterleitung erfüllt. Ausserdem erforschen wir, wie eine solche FIA in der Praxis umgesetzt werden kann.

Diese Dissertation besteht aus zwei Teilen: der erste Teil untersucht das Design einer FIA, die zwischen Rechenschaftspflicht und Privatsphäre von Sendern abwägen kann; der zweite Teil richtet sich auf Einsatzstrategien einer solchen FIA.

Im ersten Teil leisten wir die folgenden drei Beiträge. Zuerst schlagen wir die Accountable and Private Network Architecture (APNA) vor, die Rechenschaftspflicht von Sendern ermöglicht und gleichzeitig die Privatsphäre der Nutzer schützt. Im zweiten Beitrag erweitern wir APNA und stärken die Privatsphäre-Garantien, welche durch APNA ermöglicht werden. Dazu definieren wir ein neues Konzept, welches die Zuordung von Datenpaketen, die über eine Netzwerkverbindung versandt wurden, durch einen Gegner erschwert. Zudem beschreiben wir einen Angriff der durch wiederholtes Senden von Paketen ermöglicht wird und mit geringem Aufwand von einem infizierten Router ausgeführt werden kann. Als Gegenmassnahme präsentieren wir einen Netzwerk-basierten Mechanismus für die Unterdrückung von wiederholt eingespielten Paketen und verbessern dadurch die Rechenschaftspflicht von Sendern.

Der zweite Teil der Dissertation untersucht zwei verschiedene Ansätze, die den Einsatz von FIAs in Realität zu ermöglichen: einen Integrationsansatz und einen Overlay-Ansatz. Der Overlay-Ansatz realisiert eine FIA durch die Errichtung eines Overlay-Netzwerkes durch Überlagerung des heutigen Internet. Der Integrationsansatz untersucht die Integration der Hauptkonzepte einer FIA in das heutige Internet. In letzterem Fall wird die FIA nicht direkt umgesetzt, sondern dient zur Erkundung von neuen, innovativen Ideen, welche durch die Starrheit der heutigen Internet Architektur beeinträchtigt werden. Zu einem späteren Zeitpunkt werden diese Ideen durch den Einsatz bekannter Technologien im Internet realisiert. Als Beispiel beschreiben wir die Umsetzung von APNAs Grundkonzepten im heutigen Internet.

Acknowledgments

This thesis was made possible by support from my advisor, colleagues, friends, and family who provided invaluable guidance, advice, and encouragement.

I would like to express my deepest gratitude to my advisor, Prof. Adrian Perrig, for his support and continuous guidance during my studies. He provided me with motivation and a push to achieve more that helped me develop as a researcher.

I would also like to thank Prof. Heejo Lee, Prof. Timothy Roscoe, and Prof. Peter Steenkiste for being on my doctoral committee and dedicating time to read and help improve this thesis.

I would also like to thank my collaborators and colleagues from the Network Security Group. Foremost, I would like to thank Chris Pappas who I have worked very closely during my entire study; I will never forget all experiences, both from work and travel, that we shared throughout our studies. I also would like to thank Dr. Pawel Szalachowski; this thesis and related publications would not have been possible without our countless meetings and discussions. I also thank my close friends Jun Han, Yao Zhang, and Daniele Asoni who helped me get through difficult periods in this long journey. Last but not least, I would like to thank other past and present members of the group: Dr. David Barrera, Cristina Basescu, Chen Chen, Laurent Chuat, Sam Hitz, Tobias Klausmann, Dr. Jonghoon Kwon, Dr. Qi Li, Steve Matsumoto, Dr. Raphael Reischuk, Benjamin Rothenberger, Takayuki Sasaki, Kathi Schuppli, Stephen Shirley, Jean-Pierre Smith, and Ercan Ucan.

Many thanks go to the colleagues at Electronic Telecommunications Research Institute (ETRI) in South Korea. I would like to thank Dr. Woojik Chun and Dr. Heeyoung Jung who planted the seed for my research by providing me with the opportunity to work on Future Internet Architecture (FIA) research and intuitions about the Internet architecture through numerous discussions. I would also like to thank Dr. Byungok Kwak, Dr. Pyungkoo Park for making my stay in ETRI enjoyable.

Finally, I would like to thank my family for all the love and support they have given me throughout this long journey. I was only able to make

it to the end because of their encouragement and support, and this work is dedicated to them.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

The Internet was initially a network among scientists for collaborative research. Back then, users of the Internet knew and trusted each other, and communication was among trusted parties [119]. However, the Internet has expanded into a global commercial network that connects diverse people with heterogeneous trust and sometimes even with conflicting interests. Today, communication often takes place between parties that do not necessarily trust each other [44]. Moreover, some parties engage in malicious activities, such as attacking other hosts or even the network itself, e.g., for personal gains or due to conflicts of interest.

The transition to an environment with heterogeneous trust has sparked debates on whether the network should consider source accountability guarantees to hold the source responsible for any traffic that it originates, and whether the network should provide functionalities to enhance user privacy.

Importance of Source Accountability. In an environment with heterogeneous trust, users want assurance that they are communicating with whom they think they are communicating. Additionally, the lack of source accountability has become a Pandora's box for Internet security. Attackers spoof their victims' addresses and launch reflection attacks exhausting their victims' resources. Address spoofing complicates identification of the attackers and renders traffic filtering ineffective, not to mention the collateral damage when incorrectly blocking benign hosts.

Importance of Privacy. Recent revelations have shown that governments perform pervasive monitoring and mass surveillance by systematically collecting users' identities and their network traffic (e.g., NSA's upstream collection program). Worse, their surveillance capabilities are rapidly advancing with big data and data mining technologies—they can collect even larger volumes of data, and analyze the data more thoroughly. Furthermore, governments are not the only entities that use such technologies; large corporations also use them to create a profile of their customers, e.g., to make a profit by creating targeted advertisements.

Despite the importance of privacy, the network does not offer any privacy enhancing mechanisms, and users are forced to rely on application-layer tools, such as Tor [56] and VPN services, to obtain privacy guarantees. These solutions are often complex to install and have undesirable side-effects, e.g., degraded communication performance.

Both source accountability and privacy are important properties for the Internet. However, today's Internet cannot support either of the two properties; an IP address is insufficient to be an identifier since it can be spoofed. At the same time, an IP address reveals information about its user; for example, IP addresses are used in trackers [106, 155] that build histories of users' Internet activities. Furthermore, even if the Internet can be modified to support both properties, supporting the two properties at the same time is challenging since they conflict—source accountability requires identification of the sender, but to guarantee privacy the sender's identity should not be revealed.

1.1 A Vision for a Better Internet

Strong Accountability. We envision an Internet that makes an unforgeable link between the identity of a sender host and the sent packets. Such a network would raise the confidence of users that they are indeed communicating with the intended peers. Furthermore, we envision an Internet that provides new services based on strong accountability. For example, the network offers a shutoff service that allows the receivers to voice their dissent against unwanted communications and suppress such communications near the senders.

Strong Privacy. We envision an Internet that provides a range of privacy-enhancing mechanisms that users can use to protect their communication. For example, the network supports end-to-end encryption at the network-layer so that users can keep their communication private, even from state-level adversaries or Internet service providers (ISPs). Moreover, the network provides means for users to hide their identity, e.g., by using pseudonyms [79] instead of IP addresses in their packets.

Moreover, we envision an Internet that allows users to choose the privacy-enhancing mechanisms based on their privacy requirements, instead of universally imposing a certain set of mechanisms. This *design for choice* is important since privacy requirements can be very diverse even for a single

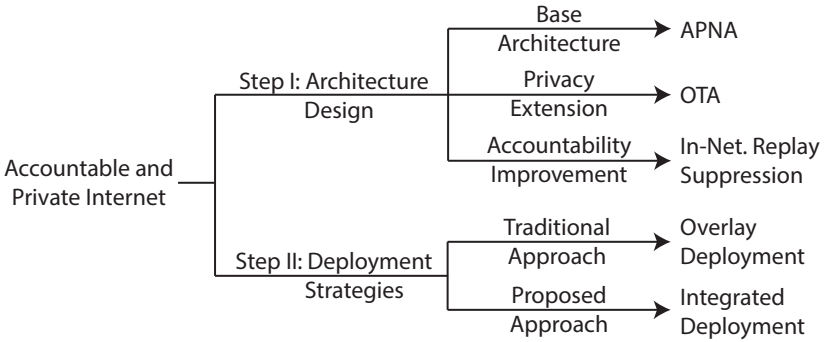


Figure 1.1: Thesis structure.

user depending on the context of communication. For example, pervasive encryption, perhaps surprisingly, may not always be desirable [51]. If a user is communicating with a mistrusting peer that may include viruses or trojans in the communication, the user may desire to have the packets sanitized by a third-party, e.g., a security middle-box, rather than hide the content via encryption.

Strong Accountability and Privacy. We envision an Internet that simultaneously provides source accountability guarantees and supports privacy-preserving communication. To this end, the inherent tension between the two properties must be resolved. To resolve the tension, for example, the sender's ISP could guarantee to a receiver that the respective sender has indeed sent the received packets, but without revealing the sender's identity to the receiver. That is, ISPs are used as the trusted third-parties to balance between privacy and accountability.

1.2 Approach

We take a two-step approach to enhance the security of the Internet in terms of source accountability and privacy (Figure 1.1). First, we design an architecture to balance between source accountability and privacy. Towards this goal, we take a clean-slate approach [37, 69] so that we can focus on the design of the architecture without constraining ourselves to the limitations or the compatibility with today's Internet.

Second, we explore deployment strategies for such an architecture. Deploying a clean-slate architecture is notoriously difficult, and there is no

definitive path to deployment. A widely-accepted approach to deploy a new architecture is to create an overlay network of the new architecture over today's Internet [143]. However, we cannot conclusively argue that this is the best approach to deploy an FIA, especially given the diversity of FIA proposals. Hence, in this dissertation, we also explore an new approach to FIA deployment.

1.3 Overview

In Chapters 2-4, we design the Accountable and Private Network Architecture (APNA) to study how to achieve both source accountability guarantees and privacy-preserving communication. In our work, our notion of source accountability protects the integrity of the source's identity and holds the source responsible for any traffic that he originates. At the same time, our notion of privacy-preserving communication allows users to control which of one's traffic can be linked to a common sender. For example, if a user sends two groups of flows and desires unlinkability between the two groups, APNA ensures that any pair of flows, one from each of the two groups, cannot be linked to a common sender.

Adversary Model. We design APNA to guarantee source accountability and protect communication privacy against an adversary whose goal is to subvert one of the two properties. We consider that an adversary has successfully broken the source accountability guarantees, if he can transmit a packet that is attributed to someone else in the network. Moreover, we consider that an adversary has successfully subverted the privacy guarantees, if he can identify a common sender across the observed traffic beyond what users have specified.

To accomplish his goals, the adversary can control any entity on the Internet except for the source host, other hosts in the same broadcast domain and the infrastructure in the source's ISP. From the entities under his control, the adversary can observe and inject packets. When injecting a packet, the adversary can create a new packet from scratch or duplicate previously observed packets. However, we assume that adversaries cannot compromise the secret keys of the ISPs and cannot break cryptographic primitives.

1.3.1 Source Accountability with Domain-Brokered Privacy

In the early stages of our efforts to design the Accountable and Private Network Architecture, we have identified the following three questions that guide our research: 1) How to balance between accountability and privacy? 2) How to guarantee source accountability? 3) How to support privacy-preserving communication?

Trust Model to Balance Accountability and Privacy. One approach to achieve the conflicting requirements of accountability and privacy is to use a trusted third-party. The third-party, serving as an accountability agent, learns the identities of the senders to attributes packets to their corresponding senders; however, it does not reveal the identities of the sender to any other entity. Then, an important design decision is to determine the entity that will serve as the trusted third-party.

There are two entities that could serve as the trusted third entity—ISPs or non-ISP entities. The proponents of the latter [38] argue that ISPs are unfit as the trusted third-party since the responsibilities of an accountability service are in conflict with their interests. For example, ISPs would be unwilling to take actions against their paying customers.

However, in our work, we enlist ISPs as the balancing point for the following reasons. Since ISPs already know the identity and the physical attachment point of their customers, they can naturally act as accountability agents. Furthermore, already on today's Internet, ISPs are encouraged and sometimes even obligated to act as accountability agents. For example, there have been efforts to urge ISPs to take actions against malicious traffic generated within their networks [53, 113, 118, 121, 156]. Additionally, some governments mandate the ISPs to keep a record of their traffic (e.g., source and destination IP addresses, packet content, etc) to identify threats against national security. At the same time, we believe that ISPs have business incentives to provide privacy features to their customers, especially in light of recent revelations regarding global surveillance.

Guaranteeing Source Accountability. The first step towards guaranteeing source accountability is to authenticate packets to their corresponding senders. One common approach to source authentication is using an access list of valid source addresses to filter packets with invalid (i.e., spoofed) source addresses. That is, a router forwards an incoming packet only if the source address in the packet is in its access list. However, the ac-

cess list-based approach has an inherent security weakness—an adversary can still spoof any of the addresses in the access list.

Instead in APNA, we use cryptographic mechanisms to authenticate every packet to avoid the security weakness of the access list-based approach. That is, the senders insert cryptographic proofs to each of the transmitted packets to attest their ownership, and their ISPs verify the cryptographic proofs to authenticate the packets.

Realizing Privacy-Preserving Communication. Today's Internet is not appropriate for privacy-preserving communications. IP addresses reveal topological and geographical information of the users; and, they can be used to link various activities of the users since, once assigned by the network, IP addresses remain unchanged for some time (e.g., until the DHCP lease timeout). Moreover, today's Internet lacks the infrastructure to support pervasive encryption at the network-layer.

In our architecture, the network provides multiple addresses, which we call ephemeral IDs (EphIDs), to users, and users use the EphIDs as they wish. For example, a user can use a different EphID for each flow or even a different one for each packet. The granularity at which EphIDs are used determines the volume of traffic that can be linked to a common sender (since those packets have the same EphID), and we leave that decision solely to the users.

We are not the first to propose the idea of using multiple addresses. Previous works [79, 133, 157] assign multiple IPv6 addresses to the hosts by creating multiple values for the interface identifiers (i.e., the lower eight bytes of the IPv6 address). These works, however, restrict the size of the anonymity set to the number of hosts within the same routing prefix. In our architecture, we design the addresses such that the anonymity set is the size of the ISP; in addition, we describe how the size of the anonymity set can be further increased.

Furthermore, we design our architecture to support end-to-end encryption at the network layer. To this end, our architecture facilitates key management by assisting two communicating hosts in establishing a shared key to encrypt their communication session.

Challenges. Although we design our architecture without confining ourselves to the limitations of today's Internet, we must still grapple with practical constraints of today's technology, such as limitations on computation

and storage resource. Specifically, we must ensure that APNA routers can forward APNA packets at a rate that is comparable to the packet forwarding rate for IPv4 packets. To this end, our source authentication protocol, which is based on cryptographic operations, must be computationally efficient. Furthermore, our architecture must avoid per-address state at the routers since multiple addresses are provided to hosts, and an excessive amount of forwarding state in routers degrades packet forwarding performance.

In addition, our architecture facilitates end-to-end encryption at the network layer. The encryption at the network layer implies that the architecture needs to assist the end hosts with key management. That is, if certificates are used for authentication, the architecture should specify the trust model for the certificates as well as specify how the network provides certificates to all hosts on the Internet. In addition, the encryption should provide perfect-forward secrecy (PFS) to prevent an adversary that obtains all long-term keys from decrypting the content of previous communication sessions.

1.3.2 Communication based on One-Time Address

Source addresses in packets can be used to link packets to a common sender, especially when an architecture provides source authentication. One way to reduce this linkability is to use different addresses for different communications. In APNA, ISPs provide multiple EphIDs to their customer hosts, and hosts decide how to use the EphIDs based on their privacy requirement. For example, a host uses a single EphID for all privacy-insensitive communication while he uses different EphIDs for each flow for privacy-sensitive communication. For highly sensitive communication, a user could choose to use a different address for each transmitted packet.

One goal of using a different address for each transmitted packet is to increase resilience against sophisticated flow-based attacks, such as traffic correlation attacks against Tor [162, 168, 169, 171, 179] and attacks that infer the content of flows by observing flow metadata (e.g., flow duration). One way to satisfy such a goal is to break the fundamental assumption of such attacks by eliminating flow information from packets so that adversaries cannot even group packets to flows. We name this property *flow-packet unlinkability* and define it as follows: by observing packets of any

number of flows, the packets are no more and no less related to any flow after the observation than they were before the observation. To this end, every packet of a flow should have a different source and destination address, and should not contain any persistent flow information (e.g., a flow identifier).

In Chapter 3, we extend APNA to support communication that achieve flow-packet unlinkability.

Challenges. In theory, APNA allows a user to use a different address for each transmitted packet; however, there are technical challenges that make APNA, as described in Chapter 2, unsuitable for such communication. First, ASes must issue EphIDs to their hosts at a significantly higher rate since the hosts use an EphID only once, to either send or receive a single packet. Additionally, to achieve flow-packet unlinkability, any flow-identifying information must be eliminated from the packets, and this leads to an interesting question: how does the communicating hosts demultiplex such packets to flows?

1.3.3 In-Network Packet Replay Suppression

According to our adversary model (Section 1.3), an adversary can replay packets of legitimate users to break the source accountability guarantees of APNA. In APNA, replayed packets are incorrectly attributed to the senders of the original packets because of APNA's cryptographic approach for source authentication. Specifically, replayed packets are exact copies of the legitimate packets, and hence contain the correct cryptographic proof.

Unfortunately, our cryptographic approach cannot be easily extended to attribute replayed packets to the replaying adversaries. In fact, designing a source authentication mechanism that can attribute replayed packets to the replaying adversary, especially one that can scale to the entire Internet, is difficult. Instead, we make a compromise and minimize the damage inflicted by replayed packets (e.g., falsely accusing the senders of the original packets for misbehaviors) by detecting and filtering replayed packets within the network.

In Chapter 4, we improve the source accountability guarantees in APNA by filtering replayed packets within the network. To this end, we first provide an additional motivation to filter replayed packets within the network by describing a new attack—router-reflection attack—to illustrate the ad-

verse consequences of replayed packets. A router-reflection attack is an attack that degrades the connectivity of a remote Internet region just by replaying packets from compromised routers on the Internet. Then, we propose a light-weight mechanism to suppress replayed packets within the network.[1]

1.3.4 Deployment Strategies for Future Internet Architectures

In Chapter 5, we consider how our architecture—APNA—can be deployed on today's Internet. Exploring new architectures that provide security properties, e.g., source accountability, is intellectually interesting, but its impact can be much greater if those ideas can be applied to today's Internet. However, deploying a new architecture is a daunting task for both technical and business reasons. Technically, updating all devices on the Internet to implement new functionalities is impossible, as the Internet is a distributed system with billions of devices. From a business perspective, incumbents on the Internet, e.g., tier-1 ISPs, are unwilling to change the Internet unless there is a critical imminent problem or they can make immediate profit [80].

To this end, we investigate two different approaches to deploy a future Internet architecture (FIA). The first deployment approach holistically deploys an FIA as an overlay over today's Internet while the second deployment approach realizes the main ideas of FIAs on today's Internet instead of deploying the FIAs themselves.

Overlay Approach to FIA deployment. In this approach, the FIA is initially deployed as an overlay over today's Internet. That is, the Internet is used as *virtual point-to-point* links to bridge islands of FIA deployments using well-established tunnel technologies (e.g., IP in IP tunnel [163], GRE tunnel [81]) to create virtual point-to-point links. Then, hope that, over time, the FIA would gradually replace today's Internet as it becomes more popular and ubiquitous. This deployment path has been described in many FIA proposals [4, 67, 75].

[1] The suppression mechanism was mainly designed by Christos Pappas, the co-author of the paper. Hence, I do not claim credit on that part of the work; however, we present the suppression mechanism for completeness of the dissertation.

Integrated Approach to FIA deployment. We explore a deployment strategy that can be summarized by the following phrase—*think revolutionary, act evolutionary*. This approach considers the design process of an FIA as an exploration of new ideas and uses the insights from the research to guide the evolution of the Internet [152]. That is, instead of concentrating the deployment effort to deploy an FIA itself, the deployment effort focuses on identifying the key ideas from the FIA and realizing them on today's Internet using well-established technologies to evolve the Internet.

Sometimes using well-established technologies cannot fully realize an idea of an FIA. For example, unicast reverse path forwarding (uRPF) [31], a well-established technology that is used to prevent source address spoofing and is already implemented in many routers, cannot fully provide the source accountability guarantees of APNA, since uRPF only provides a weak form of source authentication (See Chapter 6.2 for more detail). However, we consider using uRPF to realize the source accountability guarantees of APNA as a deployment of APNA, since we believe that such a compromise is inevitable to facilitate deployment.

We apply the integrated approach to deploy our proposed architecture, APNA. To this end, we aim to realize the ideas of APNA—source accountability guarantees and privacy guarantees—on today's Internet. As discussed above, source accountability guarantees can be realized using uRPF; hence, we focus on the latter. More specifically, we consider how to realize the notion of user-defined privacy on today's Internet.

One approach that realizes the notion of user-defined privacy is to use the privacy option of IPv6 (e.g., RFC 4941 [133]), which allows users to create multiple IPv6 addresses. Indeed, this is a viable option; however, we describe a more systematic approach to realize the notion of user-defined privacy on today's Internet.

To this end, we take a user-centric approach to privacy, and our first step is to account for users' diverse privacy requirements. Following a definition of privacy by Westin [172]—what is revealed to whom—we introduce the term of *privacy domains* that articulates what information about the user (e.g., address, packet payload) can be revealed to which entities (e.g., source, destination, and transit ISPs). As the second step, we identify privacy services that ISPs can readily deploy and that users can buy to compose privacy domains. Then, user-defined privacy is realized as fol-

lows: first, user's privacy requirements are translated to privacy domains, and these domains are composed using the identified privacy services.

1.4 Summary of Contributions

In the course of designing APNA that provides source accountability and privacy guarantees and investigating deployment strategies for APNA, we make the following high-level contributions:

- We design an architecture that balances source accountability and communication privacy. Furthermore, the architecture realizes the notion of user-defined privacy by providing two network primitives: multiple addresses to control the degree of unlinkability, encryption at the network-layer to encrypt the content of packets beyond the network-layer header.
- We introduce flow-packet unlinkability and extend our architecture to achieve this unlinkability property.
- We describe a new attack—the router reflection attack—that can be launched simply by replaying packet from compromised routers. This attack suggests that in-network duplication suppression is necessary, contrary to the conventional view that end-to-end duplicate suppression mechanisms are sufficient.
- We propose a new integrated approach to FIA deployment that aims to evolve today's Internet by realizing the main ideas of an FIA on the Internet. As an example, we apply this deployment approach to our architecture. To this end, we propose the concept of privacy domain and identify privacy services that can be used to construct privacy domains.

1.5 Related Publications

Some of the work presented in this thesis is based on the following publications, coauthored during my doctoral studies at ETH Zurich:

- Taeho Lee, Christos Pappas, David Barrera, Pawel Szalachowski, Adrian Perrig, "*Source Accountability with Domain-brokered Privacy*," in Proceedings of the ACM Conference on Emerging Networking Experiments and Technologies (CoNEXT), 2016.

- Taeho Lee, Christos Pappas, Pawel Szalachowski, Adrian Perrig, "*Communication Based on Per-Packet One-Time Addresses*," in Proceedings of the IEEE Conference on Network Protocols (ICNP), 2016.
- Taeho Lee, Christos Pappas, Adrian Perrig, Virgil Gligor, Yih-Chun Hu, "*The Case for In-Network Replay Suppression*," in Proceedings of the ACM Asia Conference on Computer and Communications Security (AsiaCCS), 2017.
- Taeho Lee, Christos Pappas, Adrian Perrig, "*Bootstrapping Privacy Services In Today's Internet*," (under submission), 2017.

Chapter 2
Source Accountability with Domain-brokered Privacy

In this chapter, we describe the base architecture of the Accountable and Private Network Architecture (APNA) that provides source accountability guarantees *and* privacy-preserving communication. In our work, our notion of source accountability protects the integrity of source's identity and holds the source responsible for any traffic that it generates. Our notion of communication privacy allows users to control which of their traffic can be linked to a common sender. To this end, we provide 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 [113, 156]. 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 communication 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 chapter, which establishes shared keys based on EphIDs, supports network-layer encryption to protect all payload data including the transport headers. A wide-spread encryption at the network layer 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 [25], 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 prohibit any further communication.

Contributions: This chapter describes 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 encryption at the network-layer with perfect forward secrecy.
- Support for a shutoff protocol that terminates unwanted communication sessions.

2.1 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.

2.1.1 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.

2.1.2 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-X* unlinkability, where 'X' could be a flow, an application, etc. We describe a concrete definition of sender-X unlinkability where 'X' is a flow to provide an intuition about sender-X unlinkability. Sender-flow unlinkability means that 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 [145].

Our second goal is data privacy through the end-to-end encryption at the network-layer. 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, such as TLS) 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 [127, p. 496].

Adversary Model: Breaking *host privacy* means that an adversary can determine the identity of a sender, or can determine if two 'X's 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 tim-

ing between packets to mitigate traffic identification based on inter-packet timing analysis [62, 82, 95, 162]. 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.

2.1.3 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.

2.2 APNA Overview

This section describes the components of our Accountable and Private Network Architecture (APNA), beginning with the role of the ASes (Section 2.2.1), followed by the use of ephemeral identifiers (Section 2.2.2), and ending with an end-to-end communication example (Section 2.2.3).

2.2.1 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 [134].

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.

2.2.2 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 [25] 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' 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 various notions of sender-X unlinkability (e.g., 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

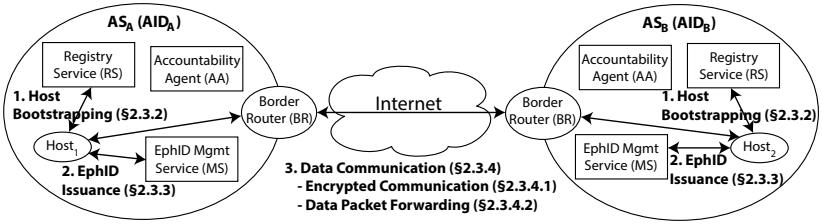


Figure 2.1: An end-to-end communication example.

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.

2.2.3 Communication Example

We describe the high-level workflow for communication between two hosts (Figure 2.1). The protocol details are provided in Section 2.3.

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.
- **EphID 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 2.1, a host in AS_A initiates communication with a host in AS_B. Communication proceeds in the following three 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. **Data Communication:** the hosts proceed with data communication using their AID:EphID tuples and their peers' AID:EphID tuples, which they obtain through out-of-band mechanisms (e.g., DNS, see Section 2.7.1). If the hosts want to encrypt their communication, they perform a connection establishment prior to the actual communication, to negotiate a shared key that will be used to encrypt their data packets. The shared key is derived from public keys that are associated with the EphIDs.

2.3 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 2.2.3). Table 2.1 summarizes the notation we use throughout the protocol description.

2.3.1 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 2.5.1.1, we describe an efficient CCA-secure encryption scheme for generating EphIDs.
- Every AS has a public key and a corresponding certificate; and there is a public-key infrastructure (e.g., RPKI [29]) from which an entity can retrieve and verify AS-certificates.

Table 2.1: 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 2.7.2.

2.3.2 Host Bootstrapping

A host authenticates to its AS using a well-established authentication protocol [63, 154]. For example, when a user subscribes to an Internet service 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 2.3.3), 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' public key (K_{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.

Once the host has successfully authenticated, the Registry Service (RS) of the AS performs the bootstrapping procedure (Figure 2.2). During this procedure, the host receives information about its AS' 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 (Line 6). Control EphIDs are used to access AS' internal services, e.g., to request data-plane EphIDs from an EphID 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 are not used for data communication. We use the term EphIDs to refer to the data-plane EphIDs.

The RS sends the host information (HID, k_{HA}) to infrastructure entities in the AS, e.g., routers, MS, AA (Line 7); the entities store the information in their database ($host\_info$) as shown in Lines 9-10. 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.

Finally, the RS returns the following information to the host (Line 8): 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.

2.3.3 Ephemeral ID Issuance

An EphID is an encrypted token using the AS' secret key (k_A). It contains the host's HID and an expiration time that indicates the validity period for the EphID. 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 the peer

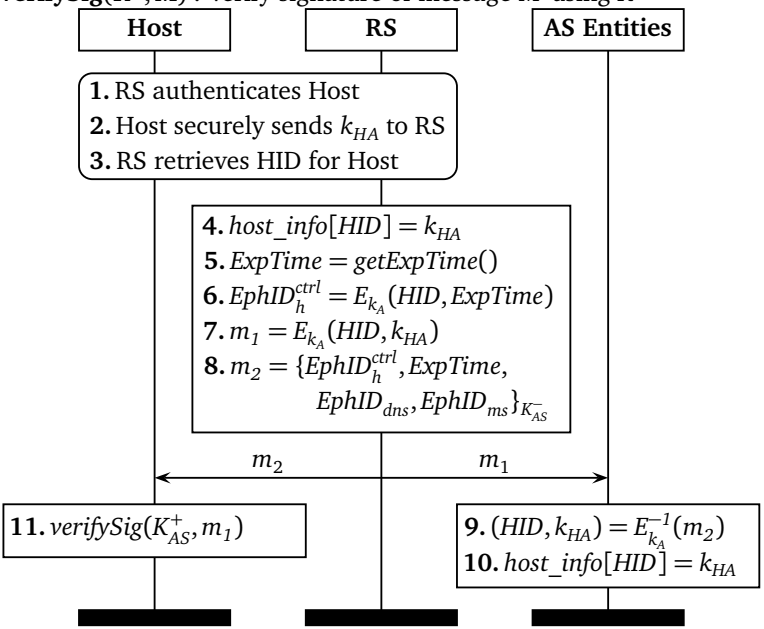


Figure 2.2: Procedure for host bootstrapping.

host, 2) to create a shared key with the peer host for data encryption (Section 2.3.4.1), and 3) to authenticate shutoff requests (Section 2.3.5). To keep the data encryption key secret from the AS, the host (not the AS) 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 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 2.3.5).

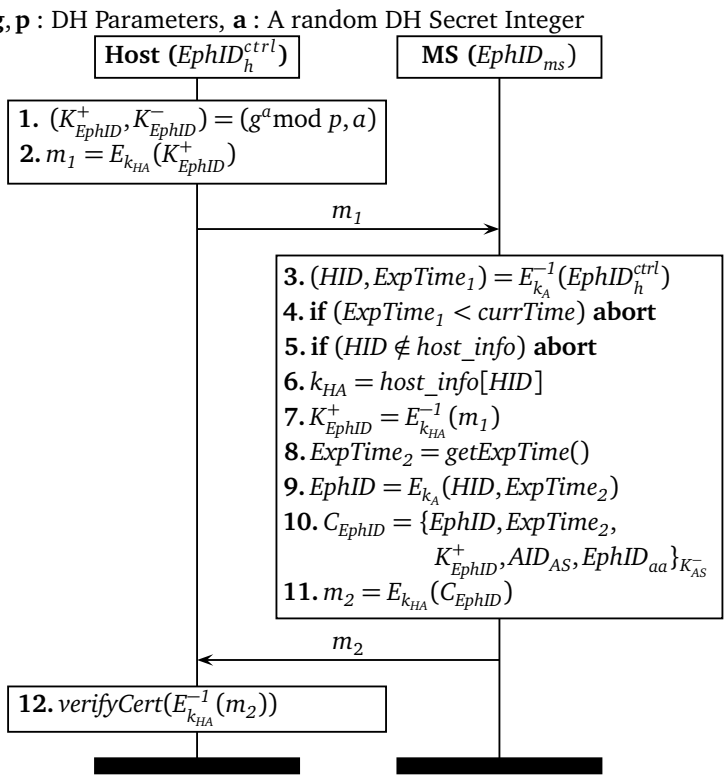


Figure 2.3: Procedure for EphID issuance.

To obtain an EphID, the host creates and sends an EphID request message to the MS (Lines 1-2 in Figure 2.3). 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-X unlinkability (see Section 2.1.2 for the adversary model) sees the content of EphID request packets, she can identify a common sender at the level of $EphID_h^{ctrl}$. Specifically, the adversary first learns the ($EphID_h^{ctrl}$, K^+_{EphID}) pair from the EphID re-

quest packets; then, the adversary sees the K^{+}_{EphID}s from the connection establishment packets (see Section 2.3.4.1), 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 $EphID^{ctrl}_{h}$. Nonetheless, she has successfully identified a common sender. 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 ($EphID^{ctrl}_{h}$) (Line 3); and performs the following checks: 1) $EphID^{ctrl}_{h}$ has not expired (Line 4), 2) the client's identifier (HID) is valid, i.e., has not been revoked (Line 5), and 3) the request message (m_1) is valid, i.e., the message can be decrypted successfully (Line 7). 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 a certificate (C_{EphID}) for the EphID (Lines 8-10). Finally, the MS encrypts the certificate and sends it to the requesting host (Line 11). The certificate is encrypted for the same reason as encrypting the EphID request packets.

2.3.4 Data Communication

In this section, we describe how hosts encrypt their communication (Section 2.3.4.1) and how APNA border routers forward data packets (Section 2.3.4.2).

2.3.4.1 Data Communication with Network-layer Encryption

Connection Establishment. For a communication that the two communicating hosts want to encrypt, the 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.

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 ob-

[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.

tained each other's EphID and the associated certificate (we discuss obtaining EphIDs through DNS in Section 2.7.1). Using the 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 connection establishment. This symmetric shared key is then used to encrypt data packets between the two hosts.

Encrypted Data Communication. Communication is based on symmetric cryptographic operations. The host first encrypts the packet with the shared key. Any existing CCA-secure encryption scheme can be used for the encryption. Then, 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 drop packets from (potentially) malicious hosts.

If the two communicating hosts do not want to encrypt their communication, then the sending host only performs the second step for the encrypted data communication; it just computes a MAC for source authentication.

2.3.4.2 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 2.4). Specifically, for external packets entering the AS, a border router checks whether the packet has arrived at the destination AS (Line 1 in the upper figure). If not, the packet is forwarded to the neighboring AS towards the destination AS (Line 8). At the destination AS, the border router checks the following conditions: 1) the destination EphID ($EphID_d$) is valid, i.e., has not expired (Line 3) and has not been revoked (Line 4), and 2) HID_D is valid, i.e., is registered and non-revoked (Line 5).

AID: AID of the Destination AS
$\mathbf{AID_S}$:$\mathbf{EphID_s}$: Source Address as AID and EphID pair in the packet
$\mathbf{AID_D}$:$\mathbf{EphID_d}$: Destination Address as AID and EphID pair in the packet
revoked_ids : List of revoked EphIDs
verifyMAC(k, M) : Verify MAC of message M using k

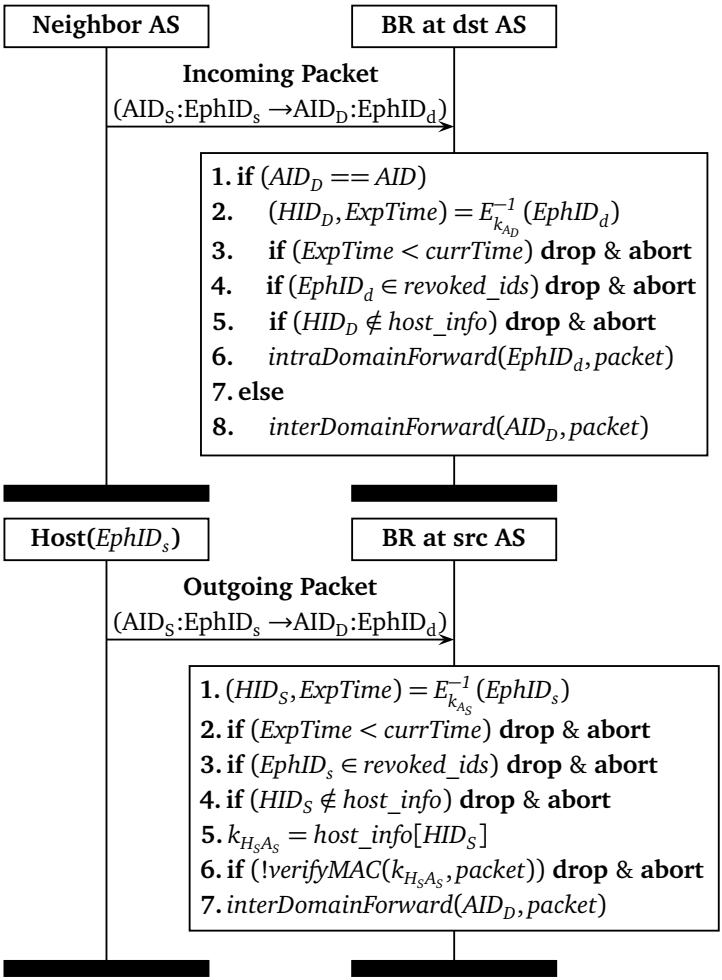


Figure 2.4: Procedures for data packet forwarding at Border Routers for incoming (top) and outgoing (bottom) packets.

If all conditions are satisfied, then the packet is forwarded to the destination host (Line 6): 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 (Line 2 in the lower figure in Figure 2.4) and has not been revoked (Line 3), 2) HID_S is valid (Line 4), and 3) the MAC in the packet is correct (Line 6).

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 (Line 5). These checks ensure that only authenticated packets leave the source AS.

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

Figure 2.5 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 (Lines 1-2): 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 $EphID_d$ (Line 3) and the signature in the request message to confirm that the re-

pkt : Packet that is sent by the Src Host but unwanted by the Dst Host
EphID$_s$, EphID$_d$: Src/Dst EphIDs in pkt
Dst : Dst Host (i.e., Host that is using $EphID_d$)
AA$_S$: Accountability Agent at Source AS

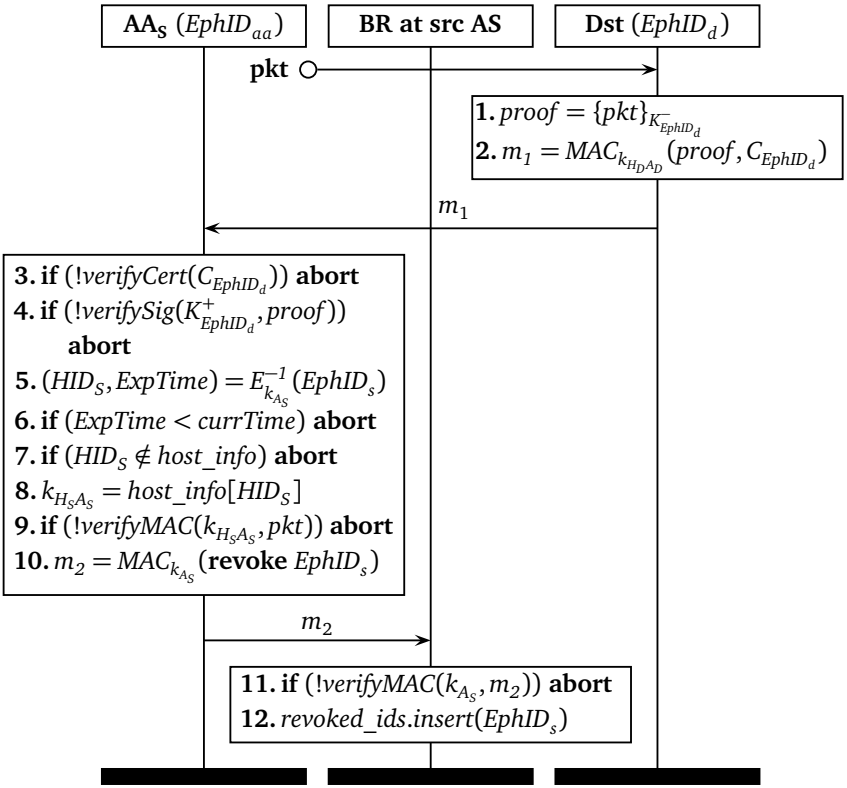


Figure 2.5: Procedure for shutoff protocol.

quest has indeed been made by D who owns $EphID_d$ (Line 4). 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 (Line 9). Finally, the AA instructs the border routers to revoke $EphID_s$ by putting it into their $revoked\_ids$ list (Line 10).

2.4 User-Defined Privacy

APNA provides two primitives that users can use to realize their privacy requirements: choice of Ephemeral IDs and encryption at the network-layer.

2.4.1 Ephemeral ID Granularity

Users can control how much information about himself can be linked from his EphIDs. That is, a user controls what 'X' would be in sender-X unlinkability by choosing how to use their EphIDs.

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 guarantees; however, additional mechanisms are necessary for the destination host to demultiplex packets into flows. In Chapter 3, we extend APNA to support per-packet EphIDs.

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.

2.4.2 Encryption at the Network-Layer

APNA supports encryption at the network-layer using the certificates associated with EphIDs. This encryption hides users' communications and makes pervasive monitoring by the state-level adversaries difficult; however, there are also disadvantages.

First, connection establishment requires one Round Trip Time (RTT) delay before any communication can take place. For communications that are not privacy-sensitive or for communications where latency is more important than data privacy, network-layer encryption may not be desired. Second, for some communications, a user may not desire to encrypt their packets to protect himself against his communicating peer. For example, if a user is communicating with an untrustworthy peer that may include viruses in the communication, the user may want to have the packets screened before he receives the packet.

Hence, APNA facilitates encryption at the network layer; however, the decision to use the network-layer encryption is left to the users.

2.5 Implementation & Evaluation

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

2.5.1 EphID Management Server

The EphID Management Server (MS) is responsible for generating EphIDs and assigning them to hosts. We describe the EphID structure, the MS implementation, and then evaluate the performance of the EphID generation procedure.

2.5.1.1 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.

We design an EphID as shown in Figure 2.6. We use 3 B for the HID, which are sufficient to uniquely represent all hosts even in large ASes. The

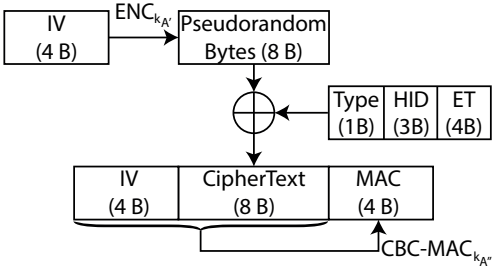


Figure 2.6: EphID construction, where ExpTime is EXP.

expiration time is 4 B long and is expressed in Unix seconds. Finally, one byte is reserved to indicate the type of the EphID. For instance, it is used to indicate whether an EphID is a control EphID or an one-time address (See Chapter 3).

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 [35] that combines a symmetric encryption with a message authentication code (MAC). 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 IV (4 B) is shorter than a single AES block (16 B) and thus the input must be padded to 16 B. We use ten rounds of AES to generate the ciphertext as recommended in RFC 3602 [73].

Next, a MAC is computed. The MAC is computed over the first 8 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 8 B of the ciphertext, 4 B IV, and 4 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).

2.5.1.2 MS Implementation

The MS generates EphIDs according to the procedure in Figure 2.3. For asymmetric cryptography, we use cryptographic primitives based on Curve25519 [39],

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 [40] and the Ed25519 SUPERCOP REF10 implementation.[3] For symmetric cryptographic operations, we leverage Intel's AES-NI encryption instruction set [76]. 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).

2.5.1.3 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 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.

2.5.2 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.

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

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

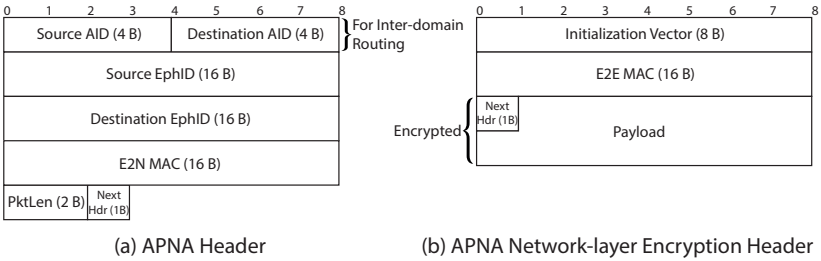


Figure 2.7: Header information.

2.5.2.1 APNA Header Information

The APNA header, shown in Figure 2.7(a), contains source and destination addresses and a MAC (i.e., E2N MAC), which is used for source authentication to the source AS, over the packet's content. More specifically, the packet consists of the following fields. The first eight bytes of the header are inter-domain addresses. We assume that AS numbers, which are four bytes, are used for inter-domain routing; in Sections 5.1 and 2.8, we show two examples of inter-domain addresses. Then, source and destination EphIDs follow, where each EphID is 16 bytes as described in Section 2.5.1.1. Next is the 16 bytes-long E2N MAC; 16 bytes is a suggested length for the Integrity Check Value (ICV) field in the Authentication Header (AH) for IPsec [124]. The last two fields are PktLen and NextHdr for which we allocate two and one byte(s), respectively. The NextHdr field indicates the header that follows the APNA header. For example, if a user uses the network-layer encryption, then the next header indicates that the APNA network-layer encryption header (Figure2.7(b)) follows the APNA header. The APNA header sums up to 59 B.

Figure 2.7(b) shows the network-layer encryption header for APNA. The header contains an initialization vector (IV) and a MAC (i.e., E2E MAC) for authenticated encryption; similar to IPsec [170], we use 8 bytes and 16 bytes for IV and E2E MAC, respectively. The encryption header also contains a next header field to indicate the header (e.g., a transport header) that follows the encryption header. All contents after the E2E MAC, including the next header field, is encrypted to hide any information beyond the network-layer.

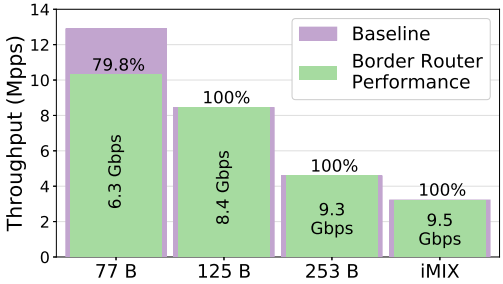


Figure 2.8: Forwarding performance.

2.5.2.2 Border Router Implementation

Our border router performs additional processing compared to traditional IPv4/IPv6 forwarding (Figure 2.4). Namely, the border router additionally performs one decryption, two table lookups, and one MAC verification. For the MAC computation, we use Galois Message Authentication Code (GMAC) based on AES since GMAC is secure against variable length input and has been proposed for data origin authentication in AH for IPsec [124].

We use DPDK [13] 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 [76].

2.5.2.3 Forwarding Performance Evaluation

We evaluate the forwarding performance on a commodity server with 10 Gbps NICs and an Intel XEON E5 CPU. To generate traffic, we use Spirent-SPT-N4U-220 [17] connected back-to-back with the server. The server receives the traffic, processes it, and sends it back to the generator.

Figure 2.8 shows the forwarding performance for multiple packet sizes and for a representative mix of Internet traffic (IMIX [129] with 368 bytes of average size); the performance baseline is forwarding without any additional processing. In APNA, a 77-byte packet is the smallest packet that consists of 14 bytes for the Ethernet header, 4 bytes for the Ethernet frame check sequence and 59 bytes for the APNA header. For 77-byte packets, performance degrades by about 20% compared to the baseline performance. However, for larger packet sizes, including the IMIX traffic, we achieve the baseline performance.

2.6 Security Analysis

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

2.6.1 Attacking Source Accountability

An adversary attacking source accountability (Section 2.1.1) 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 2.6) 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 [35].

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

[5] CBC-MAC is insecure for variable-length messages [34].

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 certificate in the request message, proving that it is an authorized party.

Relay 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.

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. In Chapter 4, we design and implement such an in-network replay detection mechanism.

2.6.2 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 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 make a detour through another AS.

2.7 Practical Considerations

2.7.1 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 and the corresponding certificate 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 EphIDs to the DNS raises a problem: a shutoff request against a published EphID would terminate any ongoing communication sessions

[6] We assume DNSSEC to authenticate DNS records.

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 2.3.4.1 is extended. To simplify the narrative, assume that the client uses $EphID_c$ to connect to the server, and 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 server using $EphID_c$ and $EphID_r$ as the source and destination EphIDs, respectively. The server verifies the certificate of $EphID_c$ and computes a shared key that will be used to encrypt data packets between the client and the server. However, the server uses the certificate of $EphID_s$—instead of the certificate of $EphID_r$—to compute the shared key. Then, in the response message to the client, the server includes the 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 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.

DNS Poisoning A malicious AS can poison its local DNS servers with rogue entries. When the victim attempts to connect to a certain domain, the AS can successfully launch a MitM attack. We do not explicitly address DNS

security since it is not a network-layer issue. With APNA, users can securely communicate with a trusted DNS server of their choice, avoiding their AS.

2.7.2 Hosts Behind Connection-Sharing Devices

In the Internet, connection sharing devices (e.g., NAT) are often used. For example, DSL or cable modems often have wireless Access Point functionality that allows multiple devices (e.g., laptops, smart phones) to connect to the Internet; and, Internet cafés make their Internet connection accessible to their customers. In this section, we describe two approaches that embrace connection-sharing devices in APNA. For brevity, a connection sharing devices is referred to as an Access Point (AP).

Bridge-mode In this approach, the AP serves as a transparent bridge that interconnects users behind the AP to the AS. The AS requires all users to be directly authenticated to itself. In this approach, the AS needs to authenticate every single user, even those that may stay in the AS network for only a short period of time.

NAT-mode In this approach, the AP creates a small AS of its own while acting as a host to the AS network. That is, the AP performs the protocol described in Section 2.3 as a host to the AS while playing the roles of a RS, an MS, a router, and an accountability agent on behalf of its clients.

As a RS, the AP bootstraps the hosts into the AP's internal network: it authenticates the hosts to the internal network, negotiates shared keys that are used to authenticate the packets that the hosts send, and provides bootstrapping information.

As an MS, the AP makes EphID requests on behalf of its hosts to the AS. The procedure that the AP follows to acquire EphIDs for its hosts is similar to the EphID issuance protocol described in Figure 2.3, but with two differences. First, when requesting for an EphID to the MS of the AS, the AP uses an ephemeral public key that is supplied by its host. Second, the AP keeps track of the EphIDs that are assigned to the hosts as a list, i.e., $EphID\_info$ (as opposed to deriving HIDs from EphIDs) since EphIDs are encrypted using the AS's secret key and EphIDs contain HIDs assigned to the AP, not to its hosts. This list is used to identify the hosts using EphIDs in the packets.

As a router, the AP implements the data forwarding procedures described in Figure 2.4, but with two differences. First, instead of parsing

the EphIDs to determine the HID of the host, the AP uses the $EphID\_info$ list. Second, for outgoing packets, in addition to verifying the MAC in the packets using the shared keys with its hosts, the AP replaces the MAC using its shared key with the AS before forwarding the packets to the AS.

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

2.8 Integration with SCION

In this section, we show that a more comprehensive security-oriented architecture can be created by combining different architectural proposals. To this end, we combine APNA and SCION [178] to construct a new architecture—SCION+APNA—that benefits from security properties of both architectures: guarantees source accountability, communication privacy, and high availability.

We choose SCION for the following reasons. First, availability is an important aspect to the Internet, and SCION is an FIA that guarantees high availability. Second, the design of APNA and SCION facilitate integration with each other. SCION focuses on inter-domain aspects (e.g., inter-domain routing) without specifying or constraining the intra-domain aspects (e.g., intra-domain routing) while APNA does the opposite; it mainly focuses on designing an intra-domain architecture.

2.8.1 SCION Background

Before introducing the SCION+APNA architecture, we provide a brief introduction to SCION; for more detailed description of SCION, we refer readers to the SCION book [141].

Internet Structure. SCION divides the ASes on the Internet into groups of independent routing sub-planes, called isolation domains (ISDs). The ASes within each ISD are organized in a hierarchical (tree-based) structure according to their provider-customer relationships, where the parent-child relationships between nodes in the trees are determined by the provider-customer relationship between ASes, with the provider AS being the par-

ent. The root of the tree, called the IDS core, consists of tier-1 ISPs that manage the ISD.

Inter-domain Routing. At a high-level, inter-domain routing in SCION consists of two steps. 1) Every AS learns a set of *path-segments* to reach its ISD core (i.e., up-segments) via path-segment construction beacons (PCBs), which are described in the next paragraph. Then, every AS registers some of the path-segments to the *Path Server (PS)* in its ISD so that a sender that wishes to communicate with a receiver in the AS can retrieve the AS' path-segments from the Path Server; such retrieved path-segments are called down-segments. 2) The source that wishes to communicate with the destination retrieves a down-segment of the destination from the path server and combines it with its up-segment to create an end-to-end communication path.[7]

PCBs are constructed as follows. The ISD core periodically initiates PCBs and sends them to its customer ASes. Then each customer AS inserts its *routing information* to the PCBs and forwards them to its customers. This process repeats until the stub ASes, i.e., leaves of the AS hierarchy, receive the PCBs.

The routing information that an AS (AS_i) inserts into a PCB contains a *hop field (HF)*, which is a pre-determined forwarding decision that its border routers will use when they forward data packets with the HF. A forwarding decision contains the following information as shown in Equation 2.1: ingress and egress interfaces of the border routers that connect to the previous and the next hop ASes, and a MAC to protect the integrity of HFs; the MAC is computed using a symmetric key that is only known to the AS.

$$HF_i = \langle InIF||EgIF||MAC_K(HF_{i-1}||InIF||EgIF)\rangle \tag{2.1}$$

For each received PCB, a stub AS learns a series of HFs, which represents the forwarding decision for the corresponding path. To send a packet on the path, a sender inserts the HFs into the packets, and border routers forward the packet based on the HFs. Hence, when a source wishes to communicate with a destination, the source 1) retrieves the down-segments of

[7] SCION also defines path-segments between ISD cores, called core-segments, and end-to-end paths are constructed by combining up-, core-, and down-segments. For simplicity, we omit core-segments in our description.

the destination from the Path Server, 2) extracts HFs from his up-segment and a down-segment of the destination, 3) combines the HFs to create an end-to-end path, and 4) embeds the combined HFs into his data packets.

2.8.2 SCION+APNA Architecture

In this section, we show the technical feasibility of SCION+APNA by describing the packet structure for the combined architecture; in Section 2.8.3, we describe a privacy implication that occurs when the two architectures are combined.

Figure 2.9 shows the structure of a data packet for SCION+APNA. The first, second, and fourth blocks (highlighted using solid boxes) of a data packet are SCION-specific headers, and they are used for inter-domain forwarding. The first block, the SCION common header, contains meta-information for inter-domain routing. Specifically, it includes a field called *CurrHF* that indicates the current hop field on the SCION Path (the fourth block in Figure 2.9) that a border router should process. Recall from Section 2.8.1 that a SCION path consists of a sequence of hop fields (HFs), where each HF is a forwarding instruction for the corresponding AS on the path. A border router in an AS must be able to identify the correct HF to forward a packet, and the router uses CurrHF for this purpose.

The SCION common header also contains additional information about the packet as follows: the SrcType and the DstType fields to indicate the intra-domain address types in source and destination ASes, respectively; packet length and the SCION header length; and, the NextHdr field to indicate the type of the header that follows the SCION header.

The third block is the APNA header, and it is used for intra-domain forwarding. The APNA header in SCION+APNA only has three fields—source and destination EphIDs, and E2N MAC—and is shorter compared to that shown in Figure 2.7(a). This is because the SCION header already includes the other information. Specifically, the SCION Path field contains information for inter-domain routing, and the SCION common header indicates the type of the next header (NextHdr) and the packet length.

The fifth block in Figure 2.9 is the APNA network-layer encryption header as indicated by the NextHdr field in the SCION common header (i.e., NextHdr=Enc). This header is defined as a SCION extension header, which is present only if the host decides to encrypt its packet. If a host does

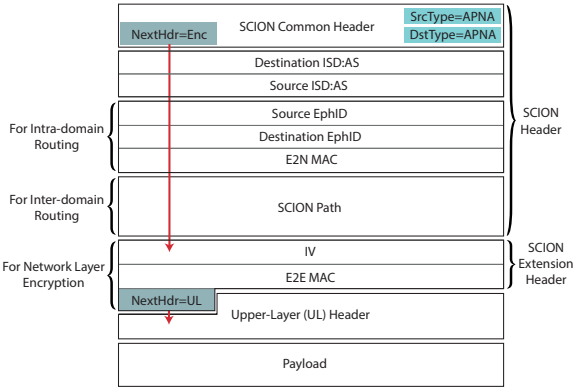


Figure 2.9: SCION+APNA packet structure.

not use the network-layer encryption, the NextHdr field would indicate the Upper-Layer header (i.e., NextHdr=UL). Lastly, the sixth and the seventh blocks are the upper-layer header (e.g., a transport layer header) and the payload, respectively. If the APNA network-layer encryption is used, these two blocks as well as the NextHdr field in the APNA encryption header are encrypted as described in Section 2.5.2.1.

2.8.3 Privacy Implication

In SCION, multiple paths are available between ASes, and this path diversity contributes to high availability. However, in SCION+APNA, this diversity creates a privacy implication. Depending on how a host chooses paths for its communications, the size of its anonymity set is reduced from the size of its AS. That is, an adversary may compromise host's desired sender-X unlinkability property.

If all hosts in an AS choose to use the same path to a destination AS, then the size of the anonymity set does not shrink. Similarly, if all hosts randomly choose a path for each of their communication, the size of the anonymity set does not change. However, if a host has an *unusual* path preference (e.g., avoid a mistrusting AS), this preference information can be used to reduce the size of the anonymity set for the host.

Figure 2.10 shows such an example. In this example, the hosts (in blue and black dots) in the bottom left AS communicate with hosts (not shown) in the other two bottom destination ASes. Furthermore, the blue host uses

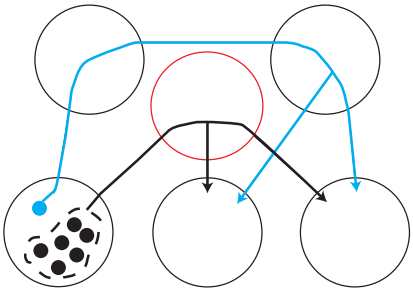


Figure 2.10: An example of privacy implication for SCION+APNA.

the longer paths (blue arrows) to avoid the red AS while all other hosts (black dots) use the shorter paths (black arrows) that go through the red AS.

Now, consider an adversary whose goal is to compromise sender-flow unlinkability and that monitors all packets leaving the bottom left AS. When the adversary observes flows to the two destination ASes through the blue paths, he can infer that these flows are more likely sent by the same sender since communications using the blue paths are unusual, thus compromising sender-flow unlinkability.

For illustration, we have used a simple example, and in reality, compromising host privacy based on hosts' path preferences would be more challenging. However, a reduction of the anonymity set is nevertheless a concern. For example, a powerful state-level adversary, which sees all traffic leaving from an AS and has enough computational capacity to run sophisticated statistical techniques (e.g., cluster analysis), may be able to reduce the anonymity set of a host based on its path preference.

2.9 Discussion

2.9.1 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) as the destination address, and one of its EphIDs (i.e., $EphID_r$) as the source EphID. Note that APNA protects

the identity of R from the source host while holding R accountable for the ICMP message.

2.9.2 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 [45]. Alternatively, the EphID issuance protocol (Section 2.3.3) 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 2.4). 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) [18].

2.9.3 Strengthening the Shutoff Protocol

If designed incorrectly, the shutoff protocol can be abused as a tool to perform DoS attacks against benign hosts. Thus, it is important to correctly identify the entities that are authorized to perform a shut-off.

In Section 2.3.5, we restricted the authorized parties as the destination host and AS since these are the only two parties that will provably receive

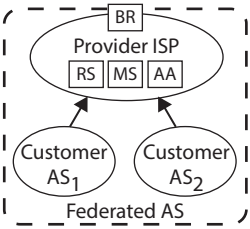


Figure 2.11: APNA-as-a-Service. The provider ISP provides APNA service to its customer ASes.

the packet based on the APNA header. However, there are proposals to encode the forwarding paths into the packets (e.g., Packet Passport [115], ICING [132], and OPT [103]). When such proposals are combined with our architecture, the list of authorized entities can be extended to include on path-ASes (or their routers), strengthening the shut-off protocol.

2.9.4 APNA-as-a-Service

An ISP can offer APNA's accountability and privacy protection not only to hosts in its network, but also to its downstream (e.g., customer) ASes as shown in Figure 2.11. In this deployment, a downstream AS can be viewed as a connection-sharing device that provides APNA connections to its hosts. Then the downstream AS can work as a *transparent bridge* or *NAT* to connect its customers to the ISP (See Section 2.7.2 for details).

APNA-as-a-Service offers benefits to both the ISP and the downstream ASes. The ISP can expand its APNA customer base beyond its network. However, note that the ISP can only offer APNA-as-a-Service to ASes whose packets must go through the ISP. This restriction is necessary since the ISP needs to be able to verify all packets that are originating from the downstream ASes to act as the accountability agent. The customer ASes, especially the small ASes that do not have a large number of hosts (i.e., small anonymity set), can enjoy stronger level of host privacy protection by mixing with customers of other (upstream) ISPs.

However, there are challenges in deploying APNA-as-a-Service. For example, the authentication process of end-hosts becomes more complicated since the hosts of the downstream AS may need to be remotely authenticated. In addition, routing in the downstream ASes become complex, especially for the ASes that are multi-homed (e.g., managing EphIDs).

2.9.5 Usability of APNA

In Section 2.4, we have described two privacy primitives—-granularity at which EphIDs are used and encryption at the network layer—that users can use to satisfy their privacy requirements. In this section, we discuss how users can use the privacy primitives for their communication.

Via the Socket API. The socket API can be extended to support privacy primitives, and one possible approach is to use the protocol argument of the `socket()` system call to express users' choice of privacy primitives. For example, `socket(AF_APNA,SOCK_STREAM,PER_FLOW_EPHID|NET_ENC)` indicates that the user would like to dedicate an EphID for the packets of the flow that would be sent using this socket and also encrypt the packets. Then, the user's operating system would assign an EphID when creating the socket, and enforce that 1) the EphID would not be used by any other flow, 2) the socket would be used only to send and receive packets that belong to the flow. In addition, the operating system would negotiate a shared symmetric key with the destination host (e.g., a server) before transmitting the user's data and encrypt packets using the negotiated symmetric key.

Via Privacy Modes in Web Browsers. Web browsers can be updated so that their privacy enhanced modes (e.g., Incognito mode in Google Chrome and Private Browsing mode in Safari and Firefox) use the two privacy primitives. For example, when a user uses the privacy mode, the browser can use EphIDs at per-session or per-flow granularity to reduce linkability between users' communication. Note that on today's Internet, user's communication can be linked to the user's IP address, even if the user uses the privacy mode. In addition, the browser can by default negotiate a shared symmetric key with each destination (e.g., servers) of the communication to encrypt communication at the network layer.

2.9.6 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.

Chapter 3
Communication Based on Per-Packet One-Time Addresses

Every field of a packet leaks some information about the communicating hosts! Adversaries ranging from WiFi stalkers to state-level agencies systematically observe packet headers and payloads in order to infer communication patterns and to obtain communication contents [27, 32, 71, 173].

Packet headers inherently leak information, since they are the foundation to achieve communication. Persistent host information across flows (e.g., source and destination addresses) is used to link flows to a common sender or a common destination, drawing a profile of communication patterns. The research community has proposed multiple solutions that provide sender-flow unlinkability. For example, an ISP-wide NAT can be used to masquerade the source address so that an adversary cannot link flows from an ISP to a common sender [149]. APIP completely removes the "source address" from the network header in order to eliminate common source information across flows [134].

Although these proposals are a step forward, packet headers in these schemes still leak valuable information. Persistent flow information across packets is used for more sophisticated attacks. For example, state-level adversaries deanonymize Tor sessions by correlating flows from the source to the Tor-entry node with flows from the Tor-exit node to the destination host [162, 168, 169, 171, 179]. Furthermore, an adversary can infer information about the content of a flow (e.g., video or VoIP traffic) by observing flow metadata (e.g., flow duration/size, and inter-packet arrival times). These attacks become feasible since host information remains the same among packets of the same flow.

In this chapter, we introduce a stronger privacy property that cannot be achieved by previous proposals—*flow-packet unlinkability*: simply by observing packets of any number of flows, the packets are no more and no less related to any flow after the observation than they were before the observation. In order to achieve this property, any flow-identifying information must be eliminated from packet headers.

Then, we extend APNA (Chapter 2) to support communications that achieve flow-packet unlinkability. To this end, we leverage on two functionalities offered by APNA. First, we make use of multiple Ephemeral IDs that are available to each host by using an ephemeral ID to send or receive exactly one packet. This also means that source addresses are not even reused as destination addresses in subsequent response packets. Second, we use the network-layer encryption offered by APNA to encrypt packet payload. The encryption prevents information leak at two levels: application-layer information (e.g., cookies) that can be used to identify hosts and link flows, and transport-layer information (e.g., explicit flow identifiers) that is used on an end-to-end basis to demultiplex packets.

In this chapter, we use a more descriptive but more concise term—*One-Time Address (OTA)*—to refer to per-packet ephemeral IDs. Furthermore, we abuse the term OTA to refer to both an address and the extended APNA architecture. However, whether the term refers to a specific address or the architecture itself should be clear from the provided context.

Contributions. We propose a holistic architecture that provides strong privacy guarantees through flow-packet unlinkability. Our architecture builds on the primitives of One-Time Addresses and network-layer encryption, and addresses the following challenges:

- flow demultiplexing at the end hosts
- efficient OTA management
  - secure and optimized OTA generation
  - no per-OTA state in the network infrastructure.
- compatibility with APNA.

3.1 Problem Setup

3.1.1 Goals

The primary goal of our architecture is to provide privacy in terms of flow-packet unlinkability. We explain this term by building on a weaker privacy property.

The first step is *host-flow unlinkability*:[1] simply by observing packets of any number of flows, the source(s) are no more and no less related to a flow

[1] Host-flow unlinkability is a more generic privacy notion than sender-flow unlinkability that refers to both the source and the destination hosts.

after the observation than they were before the observation; similarly, the destination(s) are no more and no less related after the observation than they were before the observation. That is, an adversary cannot determine if packets of two flows originate from the same host or are destined for the same host. To achieve this property, source and destination addresses need to be different for every flow, so that host-identifying information is not persistent across different flows. Note that host-related information can still be leaked at the granularity of the host's AS, e.g., by routing information or network topology: a packet can contain information that identifies the destination AS, or the ISP of a leaf AS naturally knows the source AS of all outgoing packets—it is difficult to hide this information. Our proposal also reveals flow information at the granularity of ASes.

The next step is *flow-packet unlinkability*: simply by observing packets of any number of flows, the packets are no more and no less related after the observation than they were before the observation. In other words, an adversary cannot determine if two packets belong to the same flow. To achieve this property, source(s)/destination(s) need to ensure that persistent flow-identifying information (e.g., persistent host addresses or flow identifiers) is not present across different packets.

The secondary goal of our architecture is to provide data privacy. All the exchanged content between two communicating hosts must be encrypted by default. To this end, the architecture must facilitate key management and enable hosts to negotiate cryptographic keys. Furthermore, the proposed scheme should guarantee perfect-forward secrecy (PFS) [127]: even if an adversary obtains all long-term keys of a host, he cannot subvert data privacy of past communication.

3.1.2 Threat Model

We consider two classes of adversaries who attempt to subvert our two goals, respectively.

The goal of the first adversary is to undermine flow-packet unlinkability. To achieve his goal, the adversary can: i) observe any packet in the network, including within the source and destination ASes, ii) actively inject and change packets, and iii) compromise any entity (e.g., cryptographic keys), except for the source and destination ASes.

The goal of the second adversary is to undermine data privacy by decrypting the payload of communicating hosts, which reside in two different

ASes. To achieve his goal, the adversary can: i) observe any packet in the network, including the source and destination ASes, and ii) compromise any entity (e.g., cryptographic keys), including one of the two ASes.

However, we assume that the adversaries do not perform side channel attacks (e.g., timing analysis); we believe side-channel attacks should be handled in higher layers, e.g., transport layer (See Section 3.6 for more detail.). We also assume that the cryptographic primitives we use are secure: signatures cannot be forged and encryptions cannot be broken.

3.2 Overview

Communication in our architecture is based on One-Time Addresses (OTAs). OTAs are (disposable) addresses that are issued by the Autonomous Systems (ASes) to their customer hosts; and the hosts use their OTAs as either source or destination addresses in their packets only once.

In our architecture, a host is uniquely identified by its AS through a host identifier (HID). The AS forwards packets to the host based on the HID, but the HID cannot be present in the packet headers; an adversary would correlate packets to the same host and subvert our privacy goal. Instead, hosts use OTAs that can be linked to their HIDs in the packet headers. Note that we do not impose restrictions on the form of HIDs; e.g., it can be an IP address or the hash of a public key.

In order to achieve flow-packet unlinkability, each host uses two pools of addresses: the first pool contains addresses that are associated with itself and serve as source addresses, and the second pool contains addresses of the communicating peer and serve as destination addresses. When a host sends a packet, it draws one address from each pool, to be used as a source and as a destination address, respectively. The receiving host follows the same procedure, without reusing any address in the received packet. The AS of the destination host "somehow" obtains the HID information from the destination one-time address and forwards the packet to the correct host.

Furthermore, a host can instruct its peer to supply more addresses when the address pool runs low, so that communication is not interrupted. Note that the exchanged addresses are encrypted, so that an adversary cannot infer flow information by observing address exchanges.

3.2.1 Design Space

The fundamental challenge we address is the following: Can we generate addresses that are used once, but are valid routable identifiers within an AS so that the intended recipient receives the packet? In order to justify our design decisions, we describe incremental solutions that draw a perimeter of the design space.

In a straw-man approach, the host generates an OTA on its own; and informs it to both his AS and peer host. Then, the AS stores a binding between the OTA and the HID of the host, so that it can forward an incoming packet with the generated OTA as the destination address to the correct host. Although this approach provides the desired property of flow-packet unlinkability, it comes with impractical requirements: per-packet state in the form of a mapping table from the OTA to the HID. Furthermore, this state must be distributed to all data-plane devices in the AS.

In order to reduce the impractical requirements, the host can instead encrypt its HID. Specifically, a symmetric key that is shared between all hosts and the AS can be used to generate OTAs from HIDs. In order to route a packet to a host, a forwarding device of the AS uses this shared key to decrypt the ciphertext and obtain the HID of the incoming packet; then it forwards the packet according to the HID. Since only the decryption key must be distributed among the data-plane devices in the AS, the state requirement is minimal. However, this solution provides weak security properties: one compromised host in an AS allows an adversary to compromise privacy for all other hosts in the AS, since one key is shared between all hosts and the AS.

To provide stronger security properties, a symmetric key is shared between each host and its AS: the host encrypts its HID with the symmetric key that it shares with the AS. When an incoming packet arrives, a forwarding device decrypts the ciphertext and obtains the HID. This approach introduces a circular dependency: in order to decrypt the ciphertext, the forwarding device must obtain (from the packet) a pointer to the corresponding key that was used for the encryption. However, including such a pointer in the packet breaks our privacy goal, since it introduces persistent host-identifying information across packets. In other words, the information that we try to hide must be visible to derive the shared key that was used for the encryption.

3.2.2 Key Concepts

Our approach overcomes the previously presented challenges through the combination of two concepts.

First, we use symmetric-key cryptography to moderate the excessive state requirements without sacrificing processing efficiency (use of asymmetric cryptography would sacrifice processing speed). Our evaluation (Section 3.5.1) shows that per-packet symmetric-key cryptography is efficient even on commodity machines.

Second, the AS—not the host—generates the OTAs, and it does so on the communication path: the source transmits its HID in the clear; then, the first-hop router—henceforth called access router (AR)—generates an OTA based on the HID of the host. The address is an encryption of the HID using a symmetric encryption key that is known only to the AS. When an incoming packet arrives, a forwarding device uses the symmetric key to decrypt the OTA, extracts the HID, and then forwards the packet to the destination.

Note that we obtain stricter security properties since the symmetric key is shared only among the forwarding devices in the AS, and not with the hosts of the AS. Furthermore, all OTAs are generated using one key, thus avoiding the circular dependency when obtaining this key from in-packet information.

One-Time Addresses. OTAs are the basic building block to achieve flow-packet unlinkability since they are used only once. The use of OTAs achieves the following properties.

First, they serve as privacy-preserving addresses that protect host identity. An OTA is meaningful only to the issuing AS and opaque to all other entities. That is, only the issuing AS, which already knows the identity of its customer, can identify host identities from OTAs.

Second, they serve as building blocks to achieve flow-packet unlinkability. Our architecture enables a host to use a different source OTA for every outgoing packet; and to instruct its communication peer to use a different destination OTA for every returning packet. This scheme prevents an adversary from relating packets to flows based on source and destination addresses.

OTAs, by construction, are only routable within the issuing AS, since they are opaque to any other entity. Hence, to support inter-domain com-

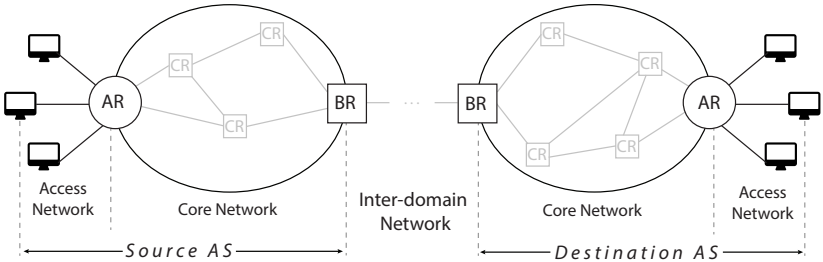


Figure 3.1: The infrastructure of an AS and the 3-layer structure of the data plane that supports OTA-based communication.

munication, packets must carry additional information about the source and destination ASes of the source and destination OTAs. Therefore, OTAs are augmented with AS Numbers (ASNs).

Adding ASN information into OTA fixes the size of the anonymity set to the number of hosts in an AS. For large ISPs with millions of customers, the size of the anonymity set is sufficiently large [150]. In Section 2.9, we have shown how to enlarge the anonymity set for small ISPs.

3.2.3 Architectural Components

We now describe the structure of the ASes and of the data plane (Figure 3.1).

Autonomous Systems. ASes play an integral role in our architecture, as they facilitate private communication for their customers. First, ASes facilitate flow-packet unlinkability by issuing OTAs to customer hosts. Second, ASes enable data privacy by acting as Certificate Authorities. Specifically, ASes issue certificates that bind OTAs to OTA-specific public keys. The associated public keys are used by the communicating hosts to negotiate shared symmetric keys that are used to encrypt the communication data.

In order to provide the required functionalities, we consider the following infrastructure components in an AS:

- Access Router (AR): connects hosts to the core network of the hosting AS. Furthermore, the AR generates and translates OTAs on behalf of its hosts.
- Border Router (BR): interconnects the core networks of different ASes. It forwards: i) outgoing and transit inter-domain traffic based

on AS information in packet headers, and ii) incoming intra-domain traffic based on host information that is extracted from OTAs.

- Core Router (CR): forwards packets in the core network of an AS, between ARs and BRs. It forwards traffic in the same way as a BR.

Data Plane. We abstract the data plane into three layers, and each layer uses different information to forward packets. At the highest layer, the network is an interconnection of ASes and packets are forwarded by border routers (BRs) using AS information (e.g., ASNs).

Next, we divide the data plane of an AS network into the core network and the access network. In the core network, packet headers contain OTAs to offer privacy guarantees; but in the access network, packet headers contain the uniquely assigned HIDs of the hosts. We discuss the security implications of our approach in Section 3.5.2.

3.3 Protocol Design

We present in detail the required steps so that two hosts (H_1 and H_2) can communicate; we present the steps in an increasing order of complexity.

In our architecture, a flow F between two hosts H_1 and H_2 is identified as FID_1 and FID_2 by each host, respectively; FIDs are used to demultiplex flows within a host, and each host chooses its own FIDs. Table 3.1 summarizes the notation that we use throughout the section.

3.3.1 Assumptions

We make similar assumptions as APNA (Section 2.3.1); however, we make one additional assumption, i.e., the third assumption.

- We assume that the cryptographic primitives (e.g., encryption) we use are secure.
- Every AS has a public key and a corresponding certificate; and there is a public-key infrastructure (e.g., RPKI [29]) from which an entity can retrieve and verify AS-certificates.
- Communication in the access network is secure. If encryption is necessary to secure the network, hosts can establish secure sessions (e.g., IPsec sessions) with their ARs and encrypt packet payloads.

HID_i	Host identifier assigned to host H_i
F, FID_1, FID_2	Flow F identified as FID_1 and FID_2 by hosts H_1 and H_2, respectively
$OTA^{j}_{H_i}$	j^{th} OTA generated for host H_i
$OTA^{(N)}_{H_i}$	A list of N OTAs that are generated for host H_i
C_{OTA_i}	Certificate for OTA_i
k_F	Symmetric key for flow F between two hosts
k_{F-AS}	Symmetric key for flow F that is shared between two hosts and their ASes
k_{tmp}	Symmetric key for flow F that is shared between two hosts and the AS of the connection initiating host (see Section 3.3.4)
k_{AS_i}	Symmetric key known among the infrastructure (e.g., routers) within AS_i
$E_k(\bullet)/E_k^{-1}(\bullet)$	Symmetric encryption/decryption of $\bullet$ with key k
K_E^+, K_E^-	Public, private key of entity E
$\{m\}_{K^-}$	Message m and the signature generated using the private key K^-
$\langle a \rightarrow b \vert m \rangle$	Packet with source and destination OTAs of a and b, respectively, and payload of m

Table 3.1: Summary of symbols and notation.

3.3.2 OTA Structure

An OTA encodes the following information: 1) the HID of the host to which the OTA is issued, and 2) the identifier of the flow FID for which the OTA is issued. The FID points to the flow-specific shared key that is used to encrypt the communication data. This information is then encrypted using the local secret key of the issuing AS (k_{AS}); the resulting ciphertext is an OTA (Equation 3.1).

$$OTA = E_{k_{AS}}(HID, FID) \tag{3.1}$$

Moreover, we require the encryption scheme to be CCA-secure: if an adversary modifies the OTA of a packet, the issuing AS will detect the modification and drop the packet. To achieve CCA-security, a different OTA must be produced for every invocation even if the same HID and FID are provided. We describe CCA-secure OTA generation in Section 3.4.

Furthermore, an OTA is associated with a certificate that binds the OTA of a host to a public key of the host; it is issued by the host's AS and serves two purposes. First, it certifies that the host owns the OTA. Only the OTAs that are used for connection establishment (see Section 3.3.4) require certificates; subsequent OTAs that are used for data communication do not require certificates. Therefore, only a few OTAs are associated with certificates. Second, the public key in the certificate is used to generate keys that are used for data encryption between the communicating hosts.

Specifically, an OTA certificate encodes the following information: 1) the OTA and 2) a public key (K^{+}_{OTA}) that is used to derive a symmetric encryption key for data communication. The certificate-issuance protocol is described in Section 3.3.5.

3.3.3 Packet Forwarding

Our architecture splits the data plane of an AS into two layers—the access network and the core network—with different forwarding mechanisms at each layer. We describe how ARs and CRs forward packets assuming that end hosts have established a connection.

Access Routers. Communication based on OTAs makes packet demultiplexing challenging: the network header does not include any flow-identifying information that would be needed to demultiplex packets at the receiver. In our architecture, ARs aid hosts in flow demultiplexing; we leverage ARs as they are the first/last AS-infrastructure component on the communication path.

Recall that a flow F is identified as FID_1 and FID_2 by the two communicating hosts. When the source host H_1 sends a packet to the destination host H_2 using an OTA of H_2, the pre-computed OTA encodes FID_2 that is used by H_2 to identify the flow. ARs process outgoing and incoming packets differently (Figure 3.2).

To send a packet to the destination host (H_2), the source host (H_1) includes information that is necessary for its AR (AR_1) to generate the correct

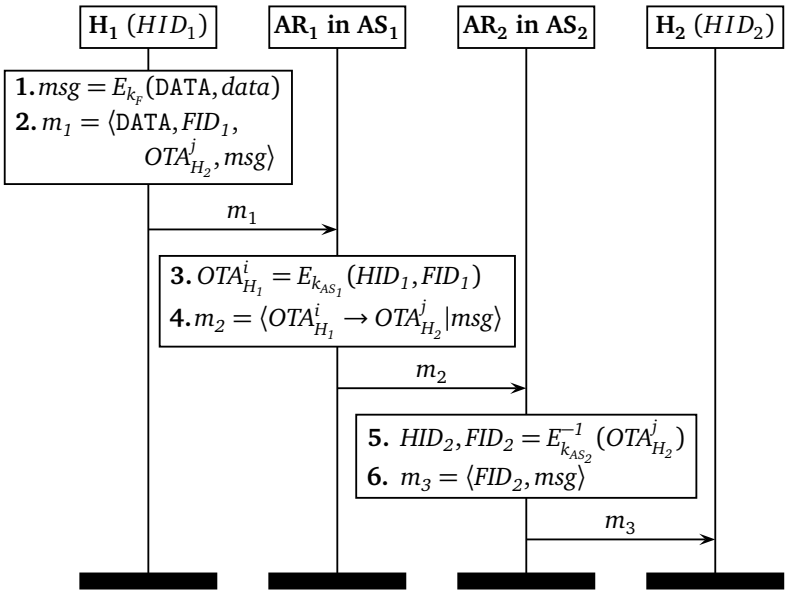


Figure 3.2: Outgoing and incoming packet processing at access routers.

OTAs (Line 2). More specifically, H_1 specifies the packet type (DATA) to indicate how AR_1 should handle the packet; the destination OTA ($OTA^j_{H_2}$); the flow identifier (FID_1) that is used by H_1 to identify the flow; and the payload (*msg*) that is encrypted using the shared key for the flow (k_F). Although flow F is identified by different identifiers for each host, there is one shared key k_F for the flow.

When AR_1 receives an outgoing packet, it performs the following tasks: it generates an OTA using H_1's identifier (HID_1) and the FID_1 in the packet (Line 3). Then using the destination OTA ($OTA^j_{H_2}$) and the payload (*msg*), the router constructs a packet and sends it to the core network (Line 4).

When AR_2 receives an incoming packet from its core network, it performs the following tasks (Lines 5-6): it extracts from $OTA^j_{H_2}$ the destination host identifier (HID_2) and the flow identifier (FID_2) that H_2 uses to identify the flow. Then, AR_2 creates a packet that includes FID_2 and the original message (*msg*); finally, it sends the packet to the destination host.

Core Routers. CRs in source ASes forward packets to BRs according to the destination ASN. CRs in destination ASes forward packets in the core network based on the destination OTAs in the packet headers. CRs do not perform address translation, but must decrypt the OTA in the packet to forward it based on the HID of the destination. This decryption is necessary since the intra-domain routing protocol is based on HIDs and since we do not want ASes to keep per-OTA state. Hence, the CRs in AS_2 perform a subset of the operations that AR_2 performs (Line 5).

3.3.4 End-to-end Communication

We have described how packets are forwarded and how flows are demultiplexed, assuming that a connection has been established. We now provide two missing pieces for fully functional end-to-end communication: connection establishment and address-pool exchange.

Connection Establishment. Connection establishment includes three main steps: 1) the initiating host authenticates the listening host, 2) the hosts negotiate shared keys for data encryption, and 3) each host informs the other of an OTA that it can use for the next packet.

Initially, the listening host waits for incoming connections similar to a traditional socket that is binded to a port. Consider a listening host H_2 that waits for packets with a specific FID (e.g., FID_2'). The initiating host has obtained an OTA of the listening host (e.g., through DNS), which contains FID_2'. The destination accepts the connection and generates a new FID_2 that will be used henceforth to identify the new flow.

The first objective of connection establishment, i.e., host authentication, is achieved through the OTA certificates. The OTAs of the listening host have corresponding certificates; the certificates are issued by the AS of the host and certify that an OTA is associated with a public key; the corresponding private key is only known to the host. The public/private key pairs are used to securely bootstrap communication and to negotiate symmetric encryption keys.

Figure 3.3 describes connection establishment between two hosts H_1 and H_2 that reside in domains AS_1 and AS_2, respectively. We make the following two assumptions. First, H_1 has received an OTA certificate ($C_{OTA^1_{H_1}}$) and an OTA ($OTA^2_{H_1}$), and H_2 has received two OTA certificates ($C_{OTA^1_{H_2}}$,

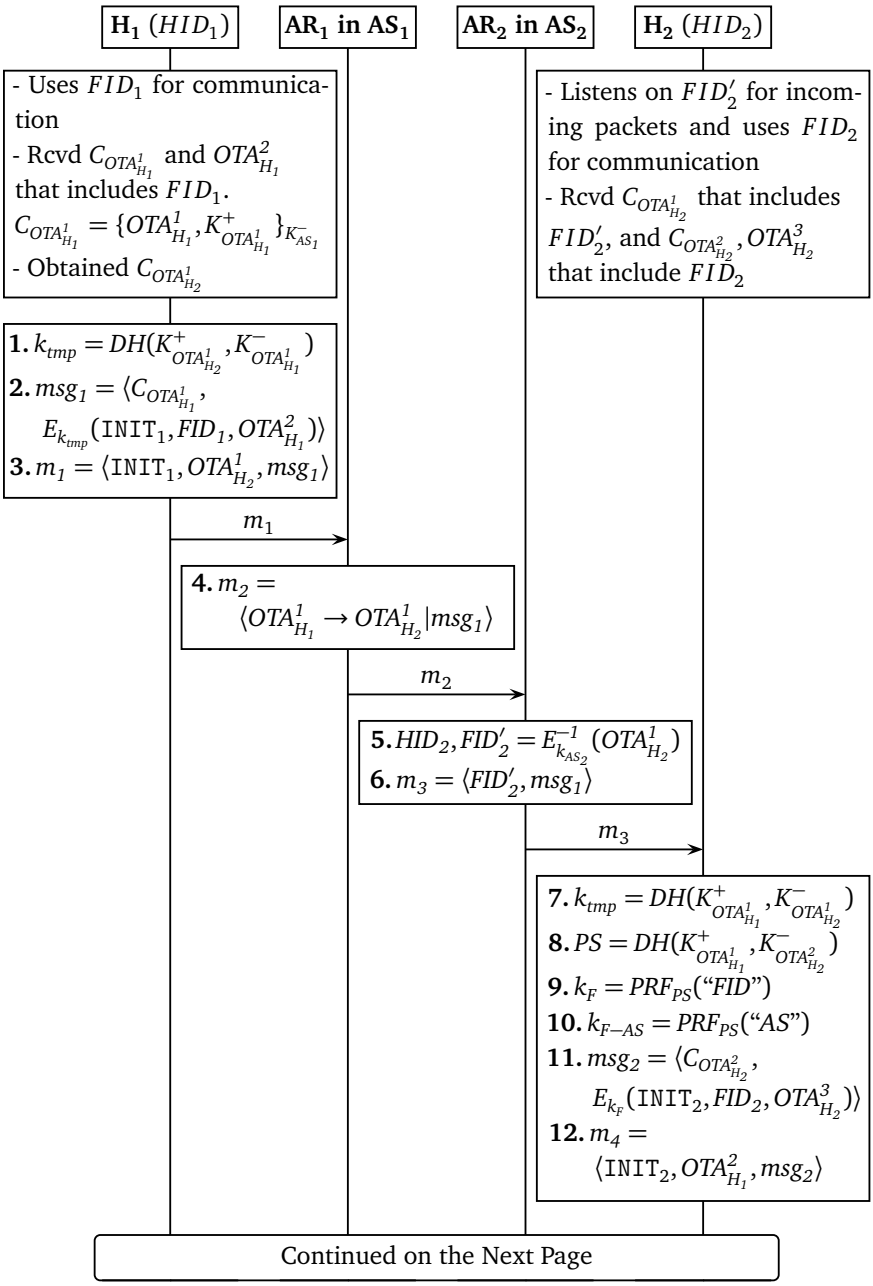
$C_{OTA^2_{H_2}}$) and an OTA ($OTA^3_{H_2}$). Second, we assume that H_1 has obtained an OTA certificate ($C_{OTA^1_{H_2}}$) of H_2, e.g., through DNS or an offline method.

Initially, H_1 generates a temporary symmetric key k_{tmp} that is used to protect the first message for the connection establishment (Line 1). Then, H_1 constructs a special connection-establishment message (msg_1). To this end, H_1 encrypts the information tuple—packet type ($\texttt{INIT}_1$), FID_1 and $OTA^2_{H_1}$, which will be used in H_2's reply packet back to H_1—using k_{tmp}, and includes the ciphertext into msg_1. The encryption protects information (i.e., FID_1 and $OTA^2_{H_1}$) that an adversary can use to compromise flow-packet unlinkability. The message (msg_1) also includes the certificate ($C_{OTA^1_{H_1}}$) whose corresponding OTA ($OTA^1_{H_1}$) is used as the source address by AR_1 (Line 4). This certificate will be used to generate symmetric keys between H_1 and H_2 for data encryption. Finally, H_1 constructs a message (m_1) for AR_1 (Line 3) that includes the packet type ($\texttt{INIT}_1$) and $OTA^1_{H_2}$, which is used as the destination address by AR_1.

AR_1 identifies that the incoming packet is used for connection establishment based on $\texttt{INIT}_1$ in m_1. AR_1 constructs a packet (m_2) using $OTA^1_{H_1}$ and $OTA^1_{H_2}$ as the source and the destination addresses, respectively, and msg_1 as the payload (Line 4); and, AR_1 sends the generated packet through the core network towards H_2. Then, AR_2 processes the incoming packet (m_2) similar to how it would process an incoming data packet, and finally forwards the packet to H_2 (Lines 5-6).

H_2 verifies the signature of AS_1 on $C_{OTA^1_{H_1}}$ and proceeds by generating the symmetric key k_{tmp} (Line 7) to decrypt message msg_1. Then, H_2 computes a pre-shared secret (PS) using the private key $K^-_{OTA^2_{H_2}}$ that is associated with $OTA^2_{H_2}$ (Line 8) and derives a new symmetric key that is shared with H_1 (Line 9). The new symmetric key (k_F) (instead of k_{tmp}) is used to guarantee data privacy; and we use this approach to provide perfect-forward secrecy (more details in Section 3.5.2).

Furthermore, another symmetric key (k_{F-AS}) is derived from the pre-shared secret (Line 10). Unlike k_F, k_{F-AS} is also shared with the ASes of H_1 and H_2, and is used to encrypt replyOTAs that a host provides to its peer. We use different keys for exchanging replyOTAs and for data privacy so that replyOTAs are only known to the end-hosts and their ASes but data communication between two end-hosts remains private even from their ASes.



Continued on the Next Page

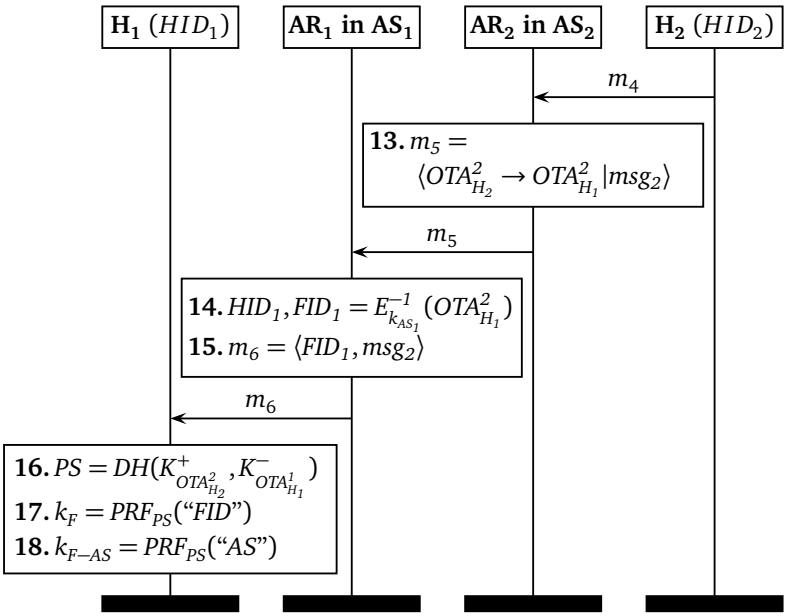


Figure 3.3: Connection establishment between two hosts.

H_2 constructs a special connection-establishment message (Line 11), which is similarly to msg_1. H_2 encrypts the information tuple—the packet type ($\texttt{INIT}_2$), the FID_2, and the reply address ($OTA^3_{H_2}$)—using k_F, and includes the ciphertext into *msg2*. In addition, H_2 also appends the certificate for the source OTA ($C_{OTA^2_{H_2}}$) to msg_2. Then, H_2 constructs a message for AR_2, which includes msg_2, packet type ($\texttt{INIT}_2$), and $OTA^2_{H_1}$ (Line 12).

AR_2 identifies that the incoming packet is used for connection establishment based on $\texttt{INIT}_2$ in m_4. Then, AR_2 constructs a packet similar to how AR_1 created m_2 and sends the generated packet through the core network towards H_1 (Line 13). AR_1 intercepts the packet and forwards it as specified by Lines 14-15 in Figure 3.3.

Finally, H_1 generates the symmetric keys k_F and k_{F-AS} using the public key in $C_{OTA^2_{H_2}}$ (Lines 16-18). Then, it obtains $OTA^3_{H_2}$ by decrypting msg_2 using k_F.

Address Pool Creation. During connection establishment, each packet carries two OTAs of the source; one that serves as a source address and one that serves as a reply address. Now, we describe a protocol that enables a host to inform its peer of valid OTAs that can be used as destination addresses in subsequent packets. This protocol is necessary since in practice there is not a one-to-one correspondence between exchanged packets; a host may send a burst of packets and therefore it needs a sufficient number of OTAs of its peer.

Figure 3.4 shows the procedure that enables a host (H_1) to request replyOTAs from its peer (H_2). H_1 creates a request by specifying a special packet type (`OTA_REQ`) and the number N of OTAs to obtain from H_2 (Lines 1-2). Then, H_1 is forwarded to H_2 in the same way as data packets.

H_2 creates a reply packet (`OTA_REP`), which will be processed by AR_2 (Line 7). The packet informs AR_2 about the number of replyOTAs to be generated and the encryption key (k_{F-AS}) that will be used to encrypt the replyOTAs. AR_2 generates N replyOTAs (Line 9), encrypts them (Line 10), and sends the packet to H_1.

The communication overhead between two hosts can be reduced by merging connection establishment and replyOTA generation. Specifically, when H_2 accepts a connection, it can instruct AR_2 to generate and attach multiple replyOTAs instead of one. With this approach H_1 does not have to request OTAs right after connection establishment.

3.3.5 Additional Functionalities

We describe two procedures that we have omitted so far.

Communication Recovery. End-to-end communication, as described so far, may result in a deadlock under certain circumstances. For example, a host may deplete its address pool of replyOTAs and the replyOTA requests may get dropped. Therefore, a recovery procedure is necessary.

The communication-recovery procedure is similar to connection establishment with one important difference: the host that initiates the recovery (e.g., H_1) must inform its peer (H_2) of which flow to resume, but since FIDs are chosen independently, H_1 must send FID_2 as the FID to be resumed. Note that we explicitly included this information in connection establishment (Lines 2 and 11 in Figure 3.3) so that both hosts know each other's FIDs.

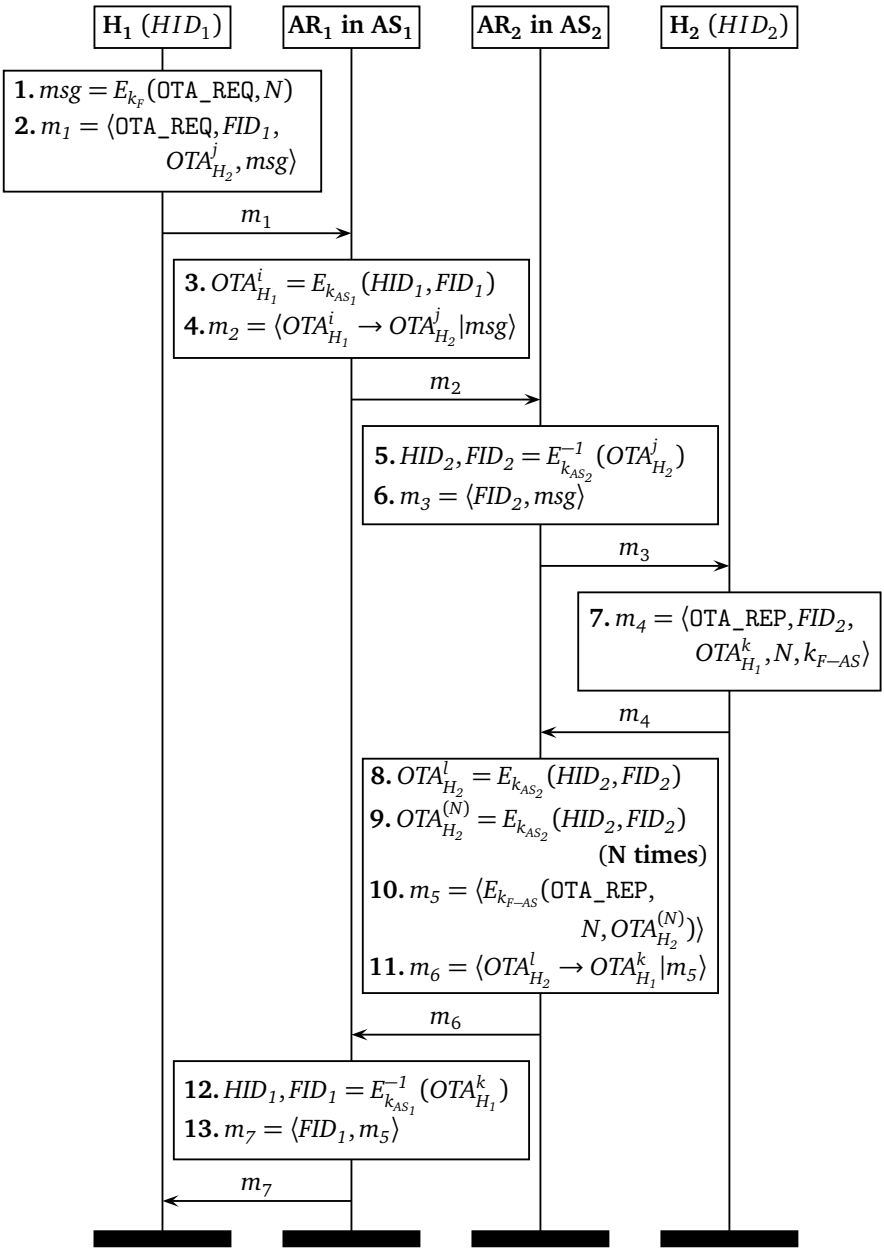


Figure 3.4: Procedure for ReplyOTA request and reply.

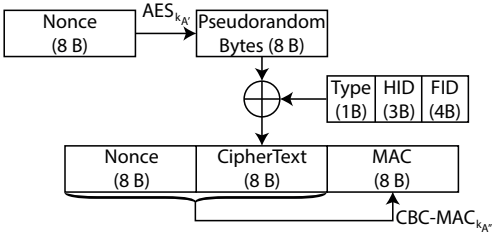


Figure 3.5: OTA specification.

Certificate Issuance. The majority of OTAs are generated on the fly during end-to-end communication. However, a few OTAs must be generated proactively as the associated certificates are used for authentication and key negotiation during connection establishment.

The procedure to generate OTA certificates is as follows. The host generates a public/private key pair (K^+, K^-) and submits K^+ and an FID to its AR. The AR issues a certificate that contains the OTA, which is generated based on the HID and FID, and the K^+. Note that K^- is never disclosed to the host's AS, which helps protecting data privacy even from the provider AS.

3.4 Implementation

We describe the implementation of our main components.

One-Time Address. We use the construction for Ephemeral IDs (Section 2.5.1.1) to construct One-Time Addresses (Figure 3.5) but with two differences. 1) An ExpTime in an EphID is replaced with a FID. 4 B is sufficient to uniquely identify all concurrent flows that a host would maintain at any given time. 2) An OTA uses 8 B each for the Nonce and the MAC fields (compared to 4 B for each field in an EphID). We use a longer nonce since we need to generate a significantly larger number of OTAs than EphIDs. In addition, we use a longer MAC to reduce the probability of collision (i.e., two different inputs resulting in a same MAC) that increases with the number of OTAs. In total, an OTA is 24 bytes.

Packet Header. Figure 3.6 shows the packet header for OTA. The header is similar to that for APNA (see Figure 2.7); however, in OTA, the meta-information (i.e., the Nonce, the E2E MAC and the NextHdr fields) for

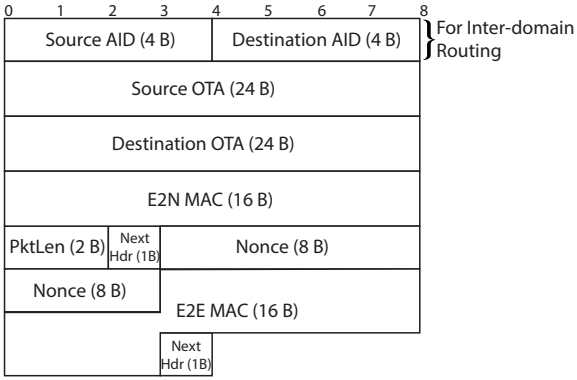


Figure 3.6: OTA header.

the network-layer end-to-end encryption is part of the packet header. This is because the encryption is essential to achieve flow-packet unlinkability, since the payloads of packets would most likely contain flow identifying information, such as a TCP header. In total, the packet header is 100 bytes.

In the packet header, there is the E2N MAC field, and thus far, we have not described the purpose of this field. This field is used to guarantee source accountability by attributing packets to their sending hosts, and we describe how this field is used in the following paragraphs.

Access Router. Access Routers are the main component of our architecture. They are responsible for generating and decrypting OTAs and issuing certificates.

ARs perform symmetric-key cryptographic operations to translate from HIDs and FIDs to OTAs and vice versa. We use hardware support (Intel AES-NI) in order to optimize these operations and guarantee a high forwarding performance.

Furthermore, ARs perform public-key cryptographic operations in order to issue certificates for OTAs (Section 3.3.5). For the public-key operations, we use the ed25519 signature scheme [40] and the ed25519 SUPERCOP REF10 implementation [14] for its high performance, short keys (32 bytes) and signatures (64 bytes).

Thus far, we have ignored source accountability, one of the two major properties of APNA. Recall from Chapter 2 that APNA defines shared symmetric keys between hosts and their ISPs, and these keys are used to

attribute packets to the corresponding senders (see Section 2.3.4.2). More specifically, a sender creates a MAC over the content of the packet using a symmetric key that he shares with his AS, and a border router of the AS verifies the MAC before forwarding the packet to the next hop AS.

In OTA, we use the access routers in the ASes, instead of the border routers, to attribute packets to their senders since we already leverage the access routers. This approach also provides the following advantages: 1) ASes do not waste their network resource by forwarding invalid packets since they would be dropped closer to the offending sources, 2) an access router only needs to maintain a fewer number of shared symmetric keys since an access router serves a fewer number of hosts than a border router, and 3) this approach reduces packet processing overhead on border routers, which need to process packets at a high rate.

End Host. A host generates a Diffie-Hellman public/private key pair (K^+, K^-) that is used to negotiate a symmetric key for data encryption during connection establishment (Section 3.3.4). In addition, K^+ becomes part of the certificate that is issued by the host's AS. We use curve25519 [39] to generate DH value pairs and use elliptic curve Diffie-Hellman (ECDH) for symmetric key negotiation.

3.5 Evaluation

We present our performance evaluation and describe the security properties of OTA-based communication.

3.5.1 Performance

We mainly focus on the performance of the AR, since it is the entity that performs all the critical functionalities that are necessary; CRs perform only a subset of this functionality. Specifically, our evaluation answers the following questions:

- Q1: How fast can ARs generate OTAs (with certificates)?
- Q2: How fast can ARs forward data packets?
- Q3: How many connection establishments per second can ARs support?

Methodology. For our evaluation, we need the following information about today's Internet: the size of access networks (in number of hosts)

and common traffic patterns in an access network, i.e., the packet rate at which hosts send/receive packets and the flow-generation rate. Due to the wide range in which these parameters can be set, we use the following conservative estimates:

- **Size of Access Networks.** In a study for CDN deployment in ISPs [88], the authors identified 1,478 distinct users over a span of 42 days; thus, we assume a typical access network with 1,500 hosts.
- **Packet Generation Rate.** We use the pricing plan of AT&T to estimate the packet rate that a user generates. For heavy Internet users, AT&T allows up to 1 TB of data every month.[2] Then, we estimate the packet-generation rate of the hosts by assuming that the hosts uniformly spend their 1 TB data allowance over a period of 30 days.[3]
- **Flow Generation Rate.** We use the CAIDA Anonymized Internet Traces Dataset [7] to estimate the flow generation rate. More specifically, we analyze a 1-hour packet trace (Equinix-Chicago monitor from 1 pm to 2 pm on 17/12/2015) and we identify a peak flow rate of 13,645 flows-per-second. Note that this number is an overestimate for our purpose for two reasons: first, the flow generation rate should be considerably lower than the flow rate; and second, the flow generation rate at an access network with 1,500 hosts will be much lower than the rate of a backbone link of a Tier-1 ISP.

We use the following settings for the evaluation:

- As described in Section 3.4, AR performs source authentication to attribute packets to their senders. To this end, AR maintains shared keys for each of the hosts in the access network; in total 1,500 symmetric keys are stored. We create a table that stores the host address (HID) as the key and the corresponding shared key as the value.
- We use a commodity desktop machine equipped with an Intel i5-3470 processor and 16 GB of DDR3 RAM to evaluate Q1.

[2] https://goo.gl/cxgwi3

[3] This is not an accurate estimate since data consumption of a host would not be uniform over 30 days; it would vary depending on the time of the day and on the day of the week. However, we could not obtain any more accurate data.

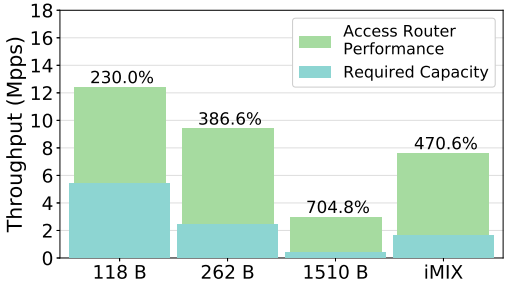


Figure 3.7: Data packet processing rate by an Access Router.

- We use the following setting to evaluate Q2 and Q3. We evaluate the forwarding performance on a commodity server with a 40 Gbps NIC and an Intel XEON E5 CPU. To generate traffic, we use Spirent-SPT-N4U-220 [17] connected back-to-back with the server. The server receives the traffic, processes it, and sends it back to the generator.

Q1: OTA and Certificate Generation. We evaluate the efficiency of OTA (Figure 3.5) and certificate generation. We generate 10^7 OTAs and their certificates and report the average generation time. On average, it takes 62 ns to generate an OTA; 63 ns to decrypt an OTA; 50.3 µs to generate an OTA and an associated certificate. The result shows that OTAs can be generated very efficiently. Although certificate generation is slower, they are only needed for communication establishment and can be generated in advance.

Q2: Data Packet Forwarding. In Section 3.3.3, we described data packet forwarding at ARs. Also recall that for outgoing packets to the core network, the AR needs to additionally attribute packets to their sending hosts. That is, for an outgoing packet, the AR: 1) retrieves the symmetric key that is shared with the sending host and authenticates the packet (i.e., ensure E2N MAC in Figure 3.6 is correct), 2) generates an OTA using the host's HID and FID in the message (m_1 in Figure 3.2). For an incoming packet from the core network, the AR decrypts the destination OTA in the packet to obtain the HID and FID information.

Figure 3.7 shows the packet processing performance for the AR. We report our results for multiple packet sizes and for a representative mix

of Internet traffic (IMIX [129] with 394 B of average size). In our architecture, the smallest packet size is 118 B that consists of 14 B Ethernet header, 4 B Ethernet frame check sequence, and 100 B OTA header. The x-axis shows the packet size and the y-axis shows the processing performance in Million-packets-per-second (Mpps). For each packet size, there are two overlapping bars. The wider bar (in blue) represents aggregate packet-generation rate for 1,500 hosts assuming that all packets have the size indicated on the x-axis. The narrower bar (in green) shows the packet processing rate by the AR. The result shows that for all packet sizes, the AR forwards packets at more than twice the required rate. Moreover, for the IMIX traffic, the AR forwards packets at 7.6 Mpps, which is 4.7 times higher than the aggregate packet transmission rate by the 1500 hosts.

CRs need to be able to forward packets at much higher rate than ARs since CRs process packets from multiple ARs. Fortunately, CRs perform fewer operations on packets than ARs—they only need to decrypt destination OTAs to identify destination HIDs, enabling CRs to forward packets at higher rate. To evaluate the packet performance of a CRs, we assume that there are 2^{20} end-hosts in an AS. That is, the CR contains a routing table with 2^{20}, where each entry consists of an HID and an outgoing port number as the key and the corresponding value, respectively.

Figure 3.8 shows the packet processing performance for the CR. We run the evaluation for the packet sizes that we used to analyze the performance of the AR. In this figure, the wider bar (in purple) represents the baseline performance without any additional processing; and the narrower bar (in green) shows the packet processing rate by the CR. For packet sizes larger than 262 B, we nearly achieve the baseline performance.

Q3: Connection Establishment Processing. To evaluate the processing rate of connection establishment packets, we need the size of a connection establishment packet (m_2 in Figure 3.3). The payload of a connection establishment packet consists of two parts—a OTA certificate and msg_1 (Figure 3.3). An OTA certificate, which consists of a OTA (28 B), a public key (32 B), and a signature (64 B), is 124 B; and msg_1, which consists of a packet type (1 B), a FID (4 B), and a reply OTA (24 B), is 29 B. Hence, the payload of a connection establishment packet is 153 B, and the packet, including the OTA header, is 253 B.

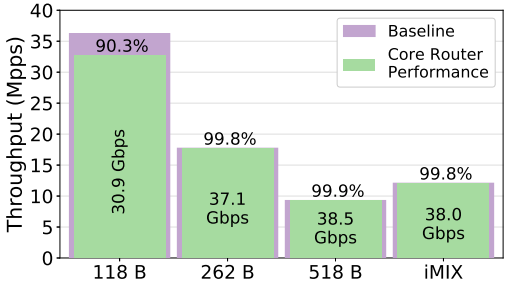


Figure 3.8: Data packet processing rate by an Core Router.

We use the AR's forwarding performance for data packets to evaluate AR's performance for processing connection establishment packets. This approach is possible because ARs perform a fewer number of operations to process connection establishment packets compared to data packets. Specifically, an AR performs source authentication for both types of packets (i.e., authenticates msg_1 in Figure 3.3 and msg in Figure 3.2); however, the AR does not generate an OTA when processing a connection establishment packet.

Figure 3.7 indicates that an AR can process connection establishment packets at an acceptable rate. The figure shows that an AR can process 262 B data packets, which is similar to the size of the connection establishment packets, at a rate of 9.39 Mpps. This rate is about 690 times higher than the peak flow rate (13,645 flows-per-second) that we observed in the CAIDA dataset.

3.5.2 Security

Compromising Data Privacy. In order to compromise data privacy, an adversary must obtain a shared key between two communicating hosts; we consider two attack scenarios.

First, we consider a MitM attack, in which the adversary impersonates a host to its peer. This attack is possible only if the adversary compromises both ASes, since the two hosts perform mutual authentication using each other's certificates that are issued by the corresponding ASes. Our threat model does not consider the compromise of two ASes, hence the attack is not possible in our setting.

Second, we consider an adversary that has captured the long-term key(s) of the host(s). Our architecture provides perfect forward secrecy (PFS), so that the adversary cannot decrypt the previous sessions, although it can decrypt the ongoing sessions. PFS is achieved since OTAs and their certificates are used once and disposed afterwards; thus the symmetric encryption key is not reused in subsequent communication sessions.

Compromising Flow-Packet Unlinkability. Since OTAs cannot be linked, packets cannot be linked to compromise flow-packet unlinkability. However, an adversary that eavesdrops on traffic in the access network compromises flow-packet unlinkability within the access network since packet headers carry HIDs. This results from the fact that OTAs are put in the packets by ARs and not the hosts themselves. An alternate approach is that hosts *prefetch all OTAs* that they will use in subsequent connections. However, this approach introduces a prohibitive bandwidth overhead, since for all outgoing packets, the host must have proactively sent another packet(s) to obtain OTA(s). We decided to place OTA generation on the communication path, sacrificing privacy for bandwidth overhead. We consider this a rational price to pay since the size of an access network is relatively small.

There is one exceptional case where an OTA may be used more than once: OTAs of public servers that register their addresses with a DNS server. This is a practical constraint since DNS cannot be updated with new OTAs for every connection of a server. However, only the destination OTA of the connection establishment is reused, which does not allow an adversary to link subsequent packets to flows.

3.6 Caveat

Eliminating flow information from packet headers directly affects traffic engineering, which consequently may have adverse effects especially for TCP performance; TCP performance is highly dependent on packet reordering. Therefore, many network devices are designed to minimize packet reordering by forwarding based on flow information in the network header (e.g., per-flow ECMP).

We do not argue that all communication sessions should be based on OTAs. Instead, we argue that the network should provide the building blocks to achieve flow-packet unlinkability, so that applications with strict privacy requirements can use it.

Chapter 4
The Case for In-Network Replay Suppression

Detecting and suppressing replayed packets in the network has been generally considered unnecessary. For example, the end-to-end argument in network design states that since an end application will detect and suppress replayed packets if deemed necessary, replay suppression is unnecessary at the network layer [158]. In this chapter, we show that, despite this seemingly persuasive argument, in-network replay detection and suppression is an indispensable network functionality for our architecture, APNA, Then we describe a highly efficient mechanism that can be used on commodity routers.[1]

We begin the chapter with the following two observations:

Router Compromises. The common assumption that routers are trustworthy no longer holds, as attackers are becoming increasingly interested and successful in compromising network infrastructure. Poor security practices [12, 15, 16] enable attackers to obtain access to routers. Even worse, the adoption of emerging technologies such as Software-defined Networking (SDN) enables attackers to compromise the network directly [6, 84].

Insufficiency of in-network source authentication in APNA. Recall from Chapter 2 that APNA uses cryptographic proofs for source authentication. More specifically, a host uses a symmetric key that he shares with his ISP to compute a MAC over a packet that he sends, and the border router of the ISP verifies the MAC to attribute the packet to the host.

This approach to source authentication is insufficient to detect replayed packets. First, since only the source ISP performs source authentication, replayed packets cannot be detected by any other entity besides the source ISP. Moreover, replay detection within the source ISP is not trivial; the MAC in a replayed packet would verify successfully, since replayed packets are identical copies of the original packet.

[1] The suppression mechanism was mainly designed by Christos Pappas, the co-author of the paper [110]. Hence, I do not claim credit on that part of the work (Sections 4.2-4.6); however, those sections are reproduced without any modification for completeness.

To detect replayed packets, one may argue that a unique counter value or a nonce could be added to each packet. However, neither approach is suitable for APNA. A nonce-based approach increases packet processing overhead at the border routers, which can only dedicate a few CPU cycles to process packets. A counter-based approach has privacy implications for communications based on one-time addresses (Chapter 3): packets that belong to a same flow are more likely to have contiguous counter values, and this information can be used to compromise flow-packet unlinkability.

To illustrate the severity of packet replays by compromised routers, we describe three adverse consequences. First, source authentication—ironic as it may sound—can help an attacker to *frame* an innocent source. For example, a compromised router can deliberately replay packets to cause abnormally high packet rates and trigger intrusion detection systems. Here, the adversary takes advantage of typical intrusion classification rules to falsely accuse a source of misbehavior; e.g., to make it appear malicious. Such attacks are particularly insidious, since the source has no readily available recourse; e.g., traffic repudiation mechanisms require global inter-ISP cooperation [139], which is difficult to orchestrate across different jurisdictions.

Second, replaying packets can be used to deliberately waste network resources and corrupt accounting mechanisms. For instance, a system that allocates network resources (e.g., bandwidth) to authenticated sources [33] can be easily overwhelmed by replaying authentic packets. Furthermore, to increase billable traffic on one of its underutilized paths, a malicious network (e.g., Tier-1 ISP) could compromise a router in an upstream network, replay authenticated traffic there, and then charge its customers for the artificially generated extra traffic.

Third, we show that the effects of potential attacks are not local, i.e., they do not affect only the implicated source(s). We present a new attack – the *router-reflection* attack – that enables an adversary to attack a geographic region of the Internet. The adversary uses a compromised router and leverages services that do not perform end-to-end replay detection (e.g., DNS or NTP): the attacker finds the *routing bottlenecks* of the target region [96] and replays requests whose responses will target these bottleneck links on the return path. Note that the attacker can easily find such bottleneck links as they are both pervasive and hard to remove in the cur-

rent Internet; and that these links are sufficiently provisioned only for a normal mode of operation, but not for targeted flooding [97, 167]. We dedicate Section 4.1 to the design and analysis of the attack.

In our quest to devise a practical in-network replay-suppression mechanism, we found that simple adaptations of well known end-to-end mechanisms cannot be used at the network layer: processing, storage and communication overheads, and time synchronization requirements raise numerous challenges.

Our in-network replay detection and suppression design is based on a combination of *per-interval sequence numbers* with small *rotating Bloom filters* that store observed packets for the currently active sequence-number window. Our design requires only minimal coordination between domains (the sequence-number-window update interval) and does not rely on global time synchronization. Furthermore, we optimize the protocol parameters to ensure very low overhead with respect to processing, storage, and communication latency. In fact, our software prototype demonstrates that in-network replay suppression is practical to perform even on commodity routers.

In summary, this chapter makes the following contribution:

- It illustrates unexpected attack capabilities enabled by in-network replays and evaluate their use in a new link-flooding attack.

4.1 Router-Reflection Attack

In this section, we describe the *router-reflection attack*, a new attack in which an adversary degrades, or blocks, legitimate traffic from flowing into a chosen geographic region of the Internet. The adversary controls a compromised router and replays packets in order to flood targeted links that carry the most routes into the region. The attack has similar goals as that of the Crossfire attack [97], but the strategy and the adversary's capabilities are different: it does not rely on large botnets; it focuses on responses from public servers, rather than requests to public servers; and it is feasible even with provably-legitimate (i.e., source-authenticated) traffic, rather than spoofed traffic.

4.1.1 Overview

Consider a set of hosts V, which are distributed over the Internet, and a set of hosts T inside a confined region of the Internet – the *target area* – against which the adversary launches the attack. A target area can include the hosts of a city, an organization, or even a small country. We refer to the traffic direction from V to T as the *inbound* direction and to its reverse as the *outbound* direction. The set of layer-3 links that carry a majority of routes from V to T are the *routing bottlenecks* of the target area. A routing bottleneck is different from a bandwidth bottleneck [86] in that a bandwidth bottleneck is determined by the traffic load, whereas a routing bottleneck is determined by the number of flows (source-destination pairs) that it carries. Typically, routing bottlenecks are adequately provisioned and the traffic flows do not experience degraded performance in the absence of flooding attacks. Henceforth, the term bottleneck refers to routing bottlenecks.

The goal of the adversary is to turn the routing bottlenecks of the target area into bandwidth bottlenecks and degrade the performance of as many flows as possible. To this end, the adversary compromises a router near the target area and replays observed traffic. Specifically, the adversary replays legitimate outbound requests from hosts in T to selected services of hosts in V that do not perform end-to-end replay detection (e.g., most UDP-based services). The corresponding responses from hosts in V hit the routing bottlenecks of the target area in the inbound direction and consume the bandwidth of these links (e.g., router R1 in Figure 4.1).

In its simplified version, the attack does not rely on traffic responses: a router can replay inbound traffic and hit routing bottlenecks that are located downstream (e.g., router R2 in Figure 4.1). Hence, a router can replay a larger portion of the observed flows – not only UDP-based services.

Our attack builds on intuition gained by recent work [96]: routing bottlenecks are target-area-specific, pervasive, and long-lived. Furthermore, the attack has three distinguishing characteristics.

1. It exploits the fact that services that do not perform replay detection are ubiquitous. There is an abundance of UDP-based services used for common tasks (e.g., DNS, SSDP, NTP) that will generate responses for replayed requests.

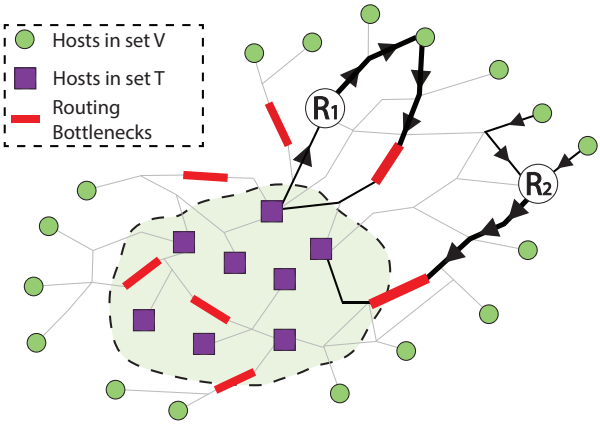


Figure 4.1: Router-Reflection Attack: compromised routers R1 and R2 can target routing bottlenecks by replaying legitimate traffic.

2. It does not inject "new" traffic nor does it modify the observed traffic. Thus, the attack does not require large botnets to create traffic, and it is feasible even with source-authentication systems [25, 103, 117] in place. Note the difference from common reflection attacks that spoof the source address, directing the response traffic to a victim.
3. It exploits the fact that Internet paths tend to be *asymmetric*, especially when they traverse core backbone links [47, 83]. This means that the responses generated by the replayed requests will likely follow a different inbound path back to the target area. Thus, a compromised router can launch such an attack without attacking itself in the inbound direction.

We emphasize that we assume a source-authentication scheme is in place. That is, a router can verify the authenticity of a packet (e.g., at the AS level) and drop modified and injected traffic. In the strict sense, source authentication should detect replayed packets as well, since the actual source of a replayed packet is the entity that injects the replayed packet. However, none of the source-authentication schemes handle in-network replay detection explicitly; this raises the new class of attacks that we describe in this section.

4.1.2 Execution

To launch a router-reflection attack against a target area, the adversary proceeds in four stages: first, she selects the set of hosts T and V; then, she computes the routing bottlenecks for the target area; next, she identifies candidate routers for compromise; and finally she uses a compromised router to replay packets of specific flows.

4.1.2.1 Stage 1: Selection of Host-Sets T and V

The adversary begins by selecting a set of public servers in a target area (set T). Furthermore, she selects a set of nodes that are geographically distributed across the globe and will act as vantage points for the target area (set V). Note that the hosts in V do not participate in the attack and are not under the adversary's control; they are used only to map the target area. The set V can be constructed using Looking Glass (LG) servers that are globally distributed. An LG server is an Internet node that is accessed remotely (usually through a web interface) and runs a limited number of commands (e.g., traceroute and ping). For instance, CAIDA provides a list with approximately 1500 LG servers located in 77 different countries and 268 different ASes [1].

4.1.2.2 Stage 2: Routing-Bottlenecks Computation

In order to compute the bottleneck links, the adversary constructs a link-map that is centered at the target area and then computes the flow density for every link in the map. We briefly present this procedure, as it has been proposed in previous work [97].

Link-Map. To construct the link-map, the adversary performs traceroutes from all vantage points (set V) to all public servers in the target area (set T), which yields $|V| \cdot |T|$ distinct traces. A trace consists of a sequence of IP addresses that belong to the interfaces of the routers on the path. The IP addresses of two adjacent routers' interfaces define a link. Thus, using all obtained traces, we get a link-map centered at the target area.

The computed link-map includes unstable routes that must be eliminated. In order to increase reliability and resource utilization, routers are often configured to load-balance their traffic over multiple paths; e.g., using per-flow or even per-packet policies [2, 3]. Thus, for the same source-destination pair, some links appear always in the traces – *persistent links*

– and some do not – *transient links*. The adversary eliminates transient links from the link-map, as they do not qualify for candidate routing bottlenecks: it is unclear whether and under which conditions replayed traffic can indeed reach a transient link.[2]

Flow-Density. Given the link-map and the traces, the adversary computes the flow density for each persistent link, i.e., the number of flows that traverse the link. A high flow density for a link means that it carries a large number of the generated traces and is an indicative metric of the overall number of flows as well.

Routing bottlenecks are determined by sorting the links in a descending order of flow density and then selecting the b highest ranked links. Higher values of b mean that more links (and thus more flows) can be considered. However, attacking only a few links is sufficient to affect a large fraction of the inbound traffic and achieve the adversary's goal.

4.1.2.3 Stage 3: Attack-Router Selection

In the third stage of the attack, the adversary discovers candidate routers for compromise. The adversary will then try to compromise routers that can target as many bottlenecks as possible.

Routers for outbound replay. The adversary discovers routers that can replay outbound traffic whose inbound responses will traverse one or more bottlenecks.

To execute this step, the adversary performs traceroutes from nodes in the target area to all hosts in V. The goal of the step is to discover as many interfaces (and thus candidate routers) as possible; thus, interfaces that perform load balancing are not eliminated. Furthermore, the adversary must perform alias resolution for the discovered interfaces, since the goal is to identify routers – not links as in the inbound direction. Note that the adversary does not control nodes in the target area, but there is a number of options to perform this step. For example, she can use an LG server that is located in the target area; or she can issue reverse traceroutes [98] to hosts in V; or use existing tools to discover the topology of an ISP [165].

Routers for inbound replay. For the simpler version of the attack, the adversary uses the traceroutes from Stage 1. Using the traces from V to T, the

[2]The traceroute dataset for our experiments (See Section 4.1.3) contains 2.3 million links, 44.6% of which are persistent.

adversary locates the interfaces (and with alias resolution the corresponding routers) that can replay packets and target bottlenecks downstream.

Our evaluation (Section 4.1.3) follows the first three stages of the attack and demonstrates that candidate routers are in the order of hundreds or thousands.

4.1.2.4 Stage 4: Packet Replay

In the final stage, the adversary has compromised one or more of the candidate routers and launches the attack. The adversary follows a similar procedure as in Stage 3, but this time using the actual observed traffic. For outbound traffic, the adversary determines which flows will result in responses that will traverse bottleneck links and ensures that she is not on the inbound path. To gain insight about the reverse path, she can use similar methods as described in Stage 3 (e.g., LG servers and reverse traceroute). For inbound traffic, the adversary must determine which of the flows can be replayed in order to target a bottleneck link that is located downstream. The adversary can simply traceroute to the destination of the flows and compare the traces with the bottlenecks computed in Stage 1. In Section 4.1.4, we discuss more practical considerations for launching the attack in both directions.

4.1.3 Experimental Results

In this section, we show that the router-reflection attack is practical; that is, we show that for a chosen target area there is an abundance of candidate routers that can be compromised to attack routing bottlenecks. Our chosen target areas {*Area1, Area2, Area3, Area4*} are a permutation of the alphabetically ordered list {*Japan, Rome, Seoul, Singapore*}. We emphasize that the feasibility and severity of the attack is not target-area specific since routing bottlenecks are an elemental property of today's Internet due to route-cost minimization [96]; thus, our findings are not limited to the above-mentioned areas.

In our experimental setup, we follow Stages 1-3 as described in Section 4.1.2. For Stages 1-2, we use approximately 200 Planetlab nodes as our vantage points, which are distributed in 34 different countries and 97 different ASes. We choose 1000 public servers in the target area using a

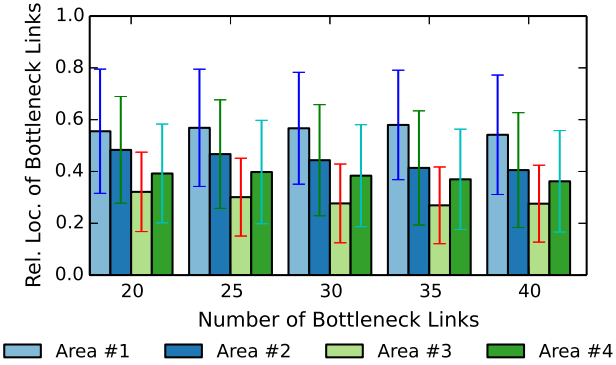


Figure 4.2: Location of bottleneck links.

public search engine with geolocation properties[3]. Furthermore, we vary the number b of links that we consider as bottleneck links from 20 to 40. For Stage 3, we choose a small number of measurement points in the target area that will be used to perform traceroutes to the vantage points; we obtain the measurement points from RIPE Atlas [5]. Finally, we perform traceroutes from all Planetlab nodes to all nodes in the target area, and from all measurement points to all Planetlab nodes. This gives us both the list of routing bottlenecks and the list of candidate routers.

Routing Bottlenecks. The first interesting result is the location of the routing bottlenecks in terms of hop distance from the vantage points and the target area. We measure the average hop distance from the source and compare it to the average path length. For many of our traces, we do not obtain responses from the last hops; usually this is due to firewalls in the hosts' local networks. In such cases, we assume that the destination resides after the last responding hop, resulting in a shorter average path length. In other words, we obtain an upper bound for the relative location of the bottlenecks with respect to the average path length.

Figure 4.2 shows the average relative location and standard deviation of the bottleneck links for each target area. The result shows that the

[3]We used SHODAN (https://www.shodan.io/) as our search engine. When target areas are more confined regions (e.g., a city), the location of the public servers must be cross-verified with other geolocation services.

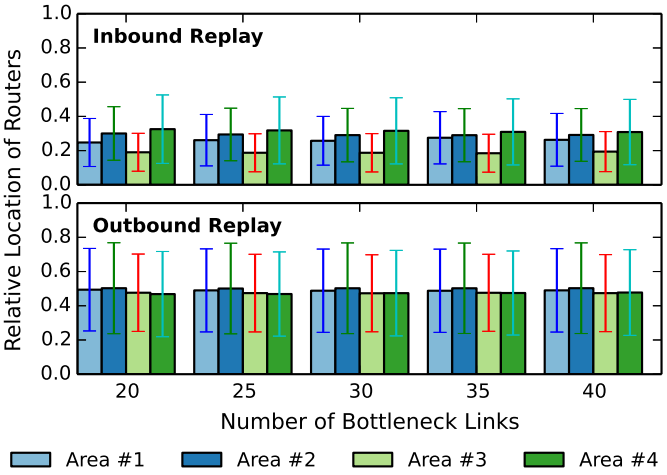


Figure 4.3: Location of routers that can target at least one bottleneck.

routing bottlenecks are located approximately in the middle of the routes and confirms the results of previous work [96]. Furthermore, the location of the links does not fluctuate significantly as the number of bottleneck links increases.

Attack Router Identification. We discover routers that can replay packets in the outbound and inbound direction, and hence, are candidates for compromise. We show the average location of candidate routers and the number of routers that can target at least one bottleneck link.

Figure 4.3 shows the average location of the candidate routers that can replay packets; the upper and lower box plots show the location of the routers for inbound and outbound replay, respectively. For inbound replay attacks, the candidate routers are located before the bottleneck links; this is expected, since bottleneck links must be located downstream with respect to the candidate routers. For outbound replay attacks, the candidate routers are located approximately in the middle of the routes; this happens because route diversity increases close to the core, and thus, routers can launch attacks without attacking themselves in the inbound direction. Again, we see no considerable change as b changes.

Figure 4.4 shows the number of candidate routers that can target at least one routing bottleneck; the upper and lower portions of each bar represent the number of candidate routers that can be used for outbound and inbound replay attacks, respectively. We observe two interesting findings. First, there is an abundant number of candidate routers to compromise, ranging from hundreds to thousands. Second, increasing the value of b does not significantly increase the number of candidate routers. This is because the additional links that are considered are adjacent to the initial routing bottlenecks, and thus, only a few more candidate routers are discovered.

4.1.4 Practical Considerations

Mitigating Measurement Inaccuracies. We had to handle two common sources of inaccuracies related to traceroutes.

First, traceroute may miss nodes, links, or even report false links [30]. We do not use specialized traceroute tools (e.g., Paris traceroute [30]), since load-balanced links cannot become routing bottlenecks. Instead, we obtain multiple traces for every flow (10 probe packets per trace) to eliminate inaccuracies due to load balancers.

Second, alias-resolution tools are mostly based on implementation specific details of routers and may introduce false negatives and false positives. False negatives fail to cluster interfaces that belong to the same router, which may reduce the number of candidates for attack routers. However, this is not an issue since we can still find plenty of candidate routers. False positives associate interfaces of different routers to the same router, which can lead to false router identification as good targets. To reduce/eliminate false positives, we use the Monotonic ID-Based Alias Resolution (MIDAR) [100] tool from CAIDA for two reasons. First, the monotonic bound tests of MIDAR yield a very low false positive rate. Second, its efficiency in resolving aliases in large lists of interface IP addresses; to resolve aliases in a list of N IP addresses it probes $O(N)$ pairs instead of testing the $O(N^2)$ candidate pairs.

Compromising Routers. In this paper, we focus on one severe consequence of router compromises rather than software security of routers; the latter is a research topic on its own right.

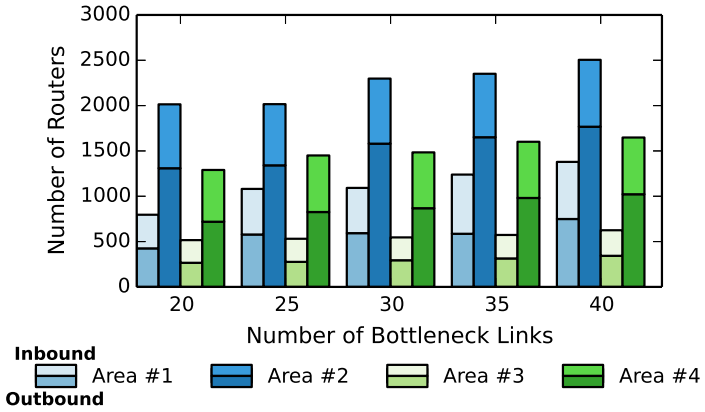


Figure 4.4: Number of routers that can target at least one bottleneck.

Our work, however, is motivated by the observation that compromised routers are already a major concern for ISPs. It is known that state-level adversaries are massively targeting routers—they are easy to compromise as they are rarely updated and lack security software to detect breaches [177]. Cisco has issued a document to warn operators of attacks against their routers and to inform them about commonly used attack vectors [50]. Security companies consider these attacks only the tip of the iceberg and highlight the difficulty of detecting such compromises, allowing attackers to maintain access for long time periods [109].

In addition, the emergence of SDN in ISP networks provides endless possibilities for infrastructure compromises, since SDN security is not yet mature. Researchers have warned that attackers can compromise networks directly [6, 84]: First, controller vulnerabilities allow attackers to compromise controllers and take control over the *entire* underlying infrastructure [147]. Second, SDN switches have proven insecure as well, allowing attackers to install persistent malware [6].

Amplification Effect. Our attack leverages UDP services since they do not perform replay suppression and they commonly have responses that are much larger than the requests that caused them; i.e., we exploit the UDP amplification effect. Recent attacks have exploited extreme amplification factors (e.g., NTP monlist commands can have a factor up to 4700) of mis-

configured services, however, moderate amplification factors are common in legitimate requests as well. For example, DNSSEC requests can have an amplification factor of 30; the GetBulk operation in SNMPv2 has an amplification factor of 6.3; and the BitTorrent hash searches have an average factor of 3.8. Routers that replay packets in the inbound direction have to rely on their available capacity to cause congestion. Note that although bottleneck links are adequately provisioned for normal network conditions, the additional load caused by small-to-moderate amplification at a router's full capacity would significantly degrade the available capacity.

Early Congestion. In case of early congestion, a link that is located upstream of the bottleneck link gets congested. Early congestion does not render the attack impossible, but confines its effect. A router that replays outbound traffic has no visibility in the inbound direction and thus cannot react to early congestion. A router that replays inbound traffic, however, has the bottleneck links located downstream. Thus, the router can perform traceroutes to the target area and determine early congestion based on the responses: in case of early congestion the router would not receive most of the ICMP replies. The router can then react by decreasing the replay rate of the corresponding flows; at the same time, it can increase the replay rate of flows that exit from the same interface, but hit another bottleneck.

Attack Detectability. A high replay rate of specific flows can trigger firewall alarms whenever network operators employ rate-limiting controls; e.g., Response Rate Limiting in DNS name servers. Although, in principle, this may limit an adversary's high-intensity replays for outbound, in practice this is easily overcome: an attack router can replay different flows that hit the same routing bottlenecks in the inbound direction.

Replaying packets in the inbound direction (see simplified attack) is less prone to rate-limiting. Intrusion detection systems and protection mechanisms are typically deployed close to the hosts and are not pervasive in the network, offering protection to resources of end systems, but not network resources. Routing bottlenecks are located in the middle of the routes and thus at a safe distance from the actual targets of the attack. This yields such defense mechanisms ineffective against inbound packet replay.

4.2 Challenges for In-Network Replay Suppression

End-to-end replay-suppression mechanisms cannot be used at the network layer due to fundamental operational differences:

1. The packet throughput of routers is orders of magnitude higher than the packet throughput of the fastest services. Consequently, a replay-detection overhead that is tolerable for a server may be intolerable for a router.
2. Routers are equipped with a few tens of MBs of fast memory (SRAM) per data-path chip. Routers use fast memory for per-packet operations in order to minimize latency and sustain a high throughput. However, hardware manufacturers do not integrate more than a few MBs of memory per chip, as the yield becomes too low to sustain the manufacturing process. On the contrary, end servers can meet their performance requirements by using larger and slower memory (DRAM) combined with the fast memory of CPU caches.
3. Novel mechanisms at the network layer often require coordination and introduce complex interactions between network entities. For example, a mechanism may require time synchronization among routers of different domains.

4.2.1 In-Network Mechanisms

To detect and suppress replayed packets in the network, a router must inspect each packet it forwards; thus, the detection mechanism must be efficient and lightweight so that it does not impair forwarding performance. We consider three main challenges for universally deploying a novel mechanism at the network layer:

1. **Computation Overhead.** In order to sustain a high throughput, a router has a strict time budget to serve a packet. The two major components that carve out this budget are latencies due to memory operations and due to CPU-intensive operations. For example, if the memory footprint for replay detection is too large to fit into

the fast memory of the router (e.g., on-chip caches), forwarding performance will be degraded due to cache misses. Likewise, if CPU-intensive operations, such as frequent public-key signature verifications become necessary, the forwarding performance will suffer.

2. **Communication Overhead.** The communication overhead can come in the form of latency overhead and bandwidth overhead. For example, if a router needs to ask another entity (e.g., the source host who created the packet or a remote router) to check the authenticity of the packet, the communication latency will increase substantially. Furthermore, the additional messages will increase the bandwidth overhead, especially if they are sent frequently.

3. **Time Synchronization.** In one extreme, a replay detection mechanism does not require any of the entities (e.g., routers, ASes, hosts) to be synchronized; and on the other extreme, it may require every entity in the Internet to be synchronized. A middle-ground solution may require only parts of the networks to be synchronized (e.g., entities within each autonomous system).

4.2.2 Inadequacy of E2E Replay Detection

Early work on replay detection identified four basic primitives, and combinations thereof, that are found in all end-to-end protocols for secure communication [120]. We present these primitives and study their applicability for in-network replay detection.

Storing Packet Digests. A router stores a digest for every packet that it forwards. When the router receives a new packet, it checks whether it has seen the packet before. This has significant advantages: it has a relatively low processing overhead (i.e., a single hash operation), no communication overhead, and no time-synchronization requirement. However, it has a large memory footprint, which makes it impractical for routers: for a fully saturated 10-Gbps link, a router needs to store 10^9 bits of data for each passing second. Even an efficient storage data structure, such as a Bloom filter is impractical: a router would need 142 MB to store packets received in one minute, assuming a target false positive rate of 10^{-5} and the largest packet size of 1500-byte, i.e., the lowest packet rate. No router can store 142 MB in its on-chip cache. Even if the router stores packets for

a minute, adversaries can still replay packets after one minute. Thus, indefinitely storing observed traffic without a mechanism to discard packets is not viable.

Sliding Time Windows. A router maintains a time window and accepts packets whose timestamps fall within the window. This requires time synchronization since the source needs to use a timestamp that falls within the time window of the router. To prevent replays, the router needs to store the packets that it has forwarded in a buffer until the packets become invalid by falling out of the sliding time window. This approach has minimal communication overhead (the timestamp in the packet) since the router does not exchange additional messages with the source. However, minimizing the size of the buffer comes at the cost of strict time synchronization so that legitimate packets are not dropped. Since precise Internet-wide time synchronization is impractical, we consider this approach also impractical.

Per-Packet Sequence Numbers. A source and an intermediate router maintain a sequence number for the source's last observed packet: the source inserts a sequence number in each packet, and the router accepts packets that have a higher sequence number than the router has previously seen from the source. This approach does not require any time synchronization, does not incur any latency overhead, and does not introduce prohibitive computation overhead. However, packet reordering can cause dropping of legitimate packets, when packets with higher sequence numbers arrive before packets with lower sequence numbers. Furthermore, this mechanism requires per-source state at routers; thus, the storage overhead depends on the granularity at which sources are identified. For example, if sources are identified at the granularity of a host, then this approach requires per-host state at routers, which is impractical.

Challenge-Response. For each packet, a router asks the source host to send a proof of transmission that verifies that the packet is not a replay: the router inserts a nonce and expects from the source a cryptographic signature over the nonce; alternately, the source can produce a message authentication code (MAC) using a key that is shared with the router. The latter approach has a relatively small computation overhead and does not require time synchronization. However, it has the largest communication overhead of all mechanisms since a separate challenge is needed for every packet that traverses every router. Hence, it is impractical.

Second Chance. This hybrid approach [120] combines three of the four primitives discussed above: it uses a variable-sized sliding time window and uses a buffer to store past packets for the duration of the window. To eliminate the requirement for time synchronization between a source and a router, it uses a challenge-response mechanism when needed: when the packet's timestamp falls outside the router's time window, the router asks the source to resend the packet using a timestamp the router provides, thereby giving the source a second chance. This approach is impractical at the router level because packets typically go through multiple routers that perform replay detection, and as a result, a packet may experience multiple rounds of second chances; thus incurring a large communication overhead that makes the approach impractical.

4.3 In-Network Replay Suppression

We present our solution for in-network replay suppression, starting with our assumption. Then, we provide a high-level overview of the solution, followed by the protocol details. Finally, we formulate an optimization problem to determine all the parameters.

Assumption: We assume that a source-authentication scheme is deployed, i.e., every packet in the network is attributed to its source. We emphasize that replay attacks are meaningful *only* if source authentication is deployed; otherwise, an adversary controlling a router can directly inject/spoof traffic and attack any target. Therefore, source authentication is a fundamental requirement of such security schemes; it is not a specific limitation of our mechanism. As we show in Section 4.1, source authentication does not prevent replay attacks, however it is necessary to prevent malicious routers from tampering with traffic.

The source-authentication literature is abundant with proposals that can be used. For instance, to authenticate packets, AIP [25] combines self-certifying IDs and an out-of-band verification protocol between an intermediate router and a source. Packet Passport [115, 117] uses multiple message authentication codes (MACs) to provide AS-AS authentication; each MAC is computed with the shared-key between the source AS and the transit AS on the path. Shared keys are generated through Diffie-Hellman key exchanges, using the public and private keys of ASes; the public keys

are obtained from RPKI [29]. Passport is a lightweight protocol, which is practical at the router level and can be used by our mechanism.

4.3.1 Overview

We build our replay-suppression mechanism based on two primitives: *per-interval sequence numbers* and *storing packet digests* in a Bloom Filter (BF). A valid sequence number is the first control check to ensure that a packet is legitimate (Section 4.3.1.1). If the packet is accepted, it is then checked against a locally stored list of previously observed packets that also have valid sequence numbers (Section 4.3.1.2). In other words: packets that are stored and replayed significantly after their observation time will be caught due to a sequence-number mismatch; packets that are replayed shortly after their observation will be caught by the BF; and packets that are replayed with a modified sequence number will be caught by source authentication. Through this combination, we build a mechanism that does not require global time synchronization and does not introduce communication overhead due to additional messages. Furthermore, our mechanism does not require adoption at every router, but can be deployed only at border routers of ASes; we discuss the security implications of the deployment locations in Section 4.5.1.

4.3.1.1 Per-Interval Sequence Numbers

Recall from Section 4.2 that the use of per-packet sequence numbers (seqNos) has two implications for a replay-suppression mechanism: the storage overhead depends on the granularity at which sources are identified, and that legitimate packets may be dropped if a packet with a higher seqNo arrives earlier than a packet with a lower seqNo.

In our approach, every source AS uses a seqNo, and every other AS (more precisely, their border router) remembers the seqNo of the source AS; in other words, routers keep per-AS state. More precisely, the source AS embeds a seqNo in every outbound packet, and transit routers only accept packets with seqNos that fall within a seqNo window. Furthermore, the source AS does not increment its seqNo per packet, but at fixed time intervals; in essence, ASes achieve loose synchronization without relying on global time synchronization. Our approach raises two important issues: the update frequency of the seqNos and the dissemination mechanism for new seqNos.

Update Frequency. The source AS periodically increments its seqNo in order to invalidate previously sent packets with smaller seqNos. Note that to achieve loose synchronization, the seqNo-update frequency is the only parameter that requires global agreement among ASes: ASes update their seqNos at a constant interval, but the seqNo values and the actual events of updating them are not synchronized. This approach makes it easier to handle packet re-ordering. A router maintains a seqNo window and only the packets with seqNos within the window are accepted. The use of per-packet seqNos makes it hard to determine an appropriate length for the window so that legitimate packets are not dropped; the length depends on parameters that change dynamically over time, such as traffic patterns (e.g., packet bursts) and load balancing at intermediate routers. With our approach we mask away this complexity, since we only need to consider the maximum variance in one-way latency (more details in Section 4.3.3).

Update Dissemination. Transit routers must be informed and keep up-to-date information for the valid seqNos of every AS, so that the observed packets can be checked. To this end, we use the following two mechanisms:

The first one is an in-band mechanism: the source AS increments its seqNo, which is carried in outbound packets, and thereby informs routers in other ASes. Upon receiving a packet, router R updates the locally stored seqNo SN_S^R for the source AS, if the packet is authentic and it carries a higher seqNo SN_S^p than the one stored. The benefit of this approach is that it does not require any additional messages to disseminate updated seqNos.

The second one is a local mechanism: a router increments the locally stored seqNo for the source AS by itself, if a new seqNo is not seen for a prolonged period of time; we refer to the time interval after which the seqNo for the source is incremented as TTL_S. This self-initiated update is necessary to limit the storage overhead at routers, since a router stores all packets with valid seqNos to prevent replays (Section 4.3.1.2); if the seqNo for a source is not updated, the router cannot delete the stored packets from the source. Thus, TTL_S will be a function of the seqNo-update interval, so that the router's seqNo for the source closely follows the source's seqNo in every case.

Table 4.1: Summary of parameters and notation.

Parameters determined by the environment	
r	Incoming packet rate at the routers.
σ	Maximum latency variation between packets.
Parameters determined by the optimization problem (§4.3.3)	
T	Sequence-number-update interval.
fp	False-positive rate.
M	Length of the sequence-number window.
L	Bloom filter switching interval.
Δ	Additional time to delay sequence-number updates.
N	Number of bloom filters.
m	Size of bloom filter.
k	Number of hash functions or bloom filter indices.
Symbols for Source AS S	
SN_S	Sequence number used by S.
Symbols for Router R	
SN_S^R	Sequence number that router R maintains for S
TTL_S	TTL until R increments SN_S^R.
BF_i	i-th bloom filter, where $0 \leq i < N$.
BF_w	Bloom filter where incoming packets are inserted.
Other Symbols	
p	An arbitrary packet.
SN_S^p	Sequence number of S that is encoded in packet p.

4.3.1.2 Storing Packet Digests

A router stores digests for previously observed packets to guarantee that packets with valid seqNos are not replayed. This is a consequence of using per-interval seqNos, since a seqNo does not uniquely identify a packet.

We create a data structure that consists of multiple BFs that are periodically rotated. In this data structure, a packet is inserted only into one of the filters, which we denote as the *writeable* filter; however, when searching if the packet has been previously observed, all filters are searched. Furthermore, the filters are periodically rotated in a round-robin fashion to prevent flooding of a single filter with too many insertions.

Algorithm 1 Processing of packets p at router R.

$authenticate(p)$	Checks if p is authentic
$get\_src\_info(p)$	Retrieves SN_S^R, TTL_S for S
$bf\_lookup(p, BF)$	Lookup p in bloom filter BF
$bf\_insert(p, BF)$	Insert p into bloom filter BF

```
1: if !authenticate(p) then
2:    return
3: end if
4: {SN_S^R, TTL_S} ← get_src_info(p)
5: if SN_S^p < SN_S^R − M then
6:    return
7: else
8:    if SN_S^p > SN_S^R then
9:       SN_S^R ← SN_S^p
10:      TTL_S ← T + Δ
11:   else
12:      for 0 ≤ i < N − 1 do
13:         if bf_lookup(p, BF_i) then
14:            return
15:         end if
16:      end for
17:   end if
18:   bf_insert(p, BF_w)
19:   Forward p
20: end if
```

We emphasize that a packet is inserted to a BF independent of the seqNo in the packet; the packet is added only to the currently active filter, the writeable filter. However, the observed packet is checked for replay by checking all BFs for membership, including the writeable one; a positive response from any of the filters indicates a replay. In order to delete packets, the filter that becomes writeable is reinitialized to zero. Note that packets in the zeroed filter have sequence numbers that are no longer valid and will be discarded if replayed. This approach naturally raises two inter-related issues: how to determine the number N of filters in the data structure and the frequency of rotation L.

Recall that a router periodically rotates the BFs, but the rotation is independent of the seqNo updates by the ASes. In order to ensure replay detection, a router must remember a packet at least until the packet seqNo is invalidated. That is, the BF coverage period must be at least as long as that of the seqNo-window, so that valid seqNos cannot be replayed. Hence, the time window that the BFs must cover, which is $N \cdot L$, must exceed the amount of time that is needed for the packet to become invalid. At the same time, BFs must be small to reduce storage requirements and fit in fast memory. In Section 4.3.3, we take N and L into account to compute the optimal parameters for our replay-suppression protocol.

4.3.2 Protocol Operations

We present the tasks that are performed by the egress border routers of the source AS S and by the ingress border routers R of the intermediate and destination ASes. Table 4.1 summarizes the parameters and the notation we use.

Source AS: S inserts its seqNo SN_S in every outbound packet. In addition, it increments its seqNo SN_S after each interval T.

Ingress Router: For each incoming packet from a neighboring AS, R checks if the packet falls within the seqNo window for S and if the packet is present in any of the BFs. Algorithm 1 describes the procedure that R executes for incoming packets.

Initially, the router checks the authenticity of the packet (Lines 1-2) and checks if the seqNo in the packet (SN_S^p) falls within the seqNo window (Lines 5-6). If the packet is not authentic or has an invalid seqNo, the packet is dropped and the procedure terminates. Then, for valid packets, the router checks if the packet has been previously recorded in any of the N BFs (Lines 12-16). If the packet has not been seen previously, it is added to the writeable filter (Line 18) and then it is forwarded (Line 19). Note that when querying/adding a packet to a BF, the router computes a keyed pseudo-random-function (PRF) over the content of the packet, excluding the mutable packet fields. The output of the PRF is used to determine the bits in the BFs that must be checked/set; the key for the PRF is known only to the AS. The use of a keyed PRF is necessary to prevent an adversary from launching a chosen insertion attack against the BFs: if the adversary can

control which bits in the writeable filter are set, it can set all bits in the BFs and cause *all* packets to be recognized as replays.

Furthermore, if the seqNo in the packet is higher than the one locally stored for S, R updates its seqNo and reinitializes TTL_S to $T + \Delta$ (Lines 8-10), which is the count-down timer used to self-update the seqNo of the source. Δ is an additional short delay that is used to ensure that the router does not increment the source's seqNo faster than the source does. If a router were to increment the source's seqNo faster, then it could be that after a long time period packets from the source AS would be dropped.

In addition, R performs one more task (not described in Algorithm 1). It periodically decrements TTL_S for each AS; if TTL_S becomes zero, the seqNo of the corresponding AS (SN_S^R) is incremented and TTL_S is reset to $T + \Delta$. This implements the count-down timer that is used for the self-updates of the sources' seqNos.

4.3.3 Optimization Problem

In this section, we formulate an optimization problem that involves the inter-related parameters of our mechanism. An appropriate configuration of the parameters and especially of the BFs is crucial to guarantee a high forwarding performance for the routers. We describe performance as a function of all the involved parameters: $f(m, k, N, L, M)$. Then, we derive constraints between the parameters, which gives us the following optimization problem:

$$\begin{aligned}
&\text{minimize} && f(m,k,N,L,M) \\
&\text{subject to} && M > \frac{\sigma}{T} + 1, && (4.1)\\
& && N > \frac{1.1(T+\sigma)}{L} + 1, && (4.2)\\
& && m > \frac{-krL}{\ln(1-(1-(1-fp)^{\frac{1}{N}})^{\frac{1}{k}})}, && (4.3)\\
& && m,k,N,L,M \in Z^+ && (4.4)
\end{aligned}$$

Sequence-number update interval (T). T represents the time period for which a seqNo is used, before it is incremented. We consider values on the

order of a few milliseconds (e.g., 10ms), and we show that it leads to an efficient implementation.

Additional delay window (Δ). Δ is a delay period that is used to slow down the update rate of router's R view for the seqNo of AS S (SN_S^R). It ensures that R does not increment its seqNo faster than S, i.e., does not increment at an interval shorter than T.

The main reason for an early update is the clock drift between S and R. We define Δ with respect to T, since the amount of clock drift is proportional to the time period under consideration; we are interested in estimating the seqNo-update inaccuracy for SN_S and SN_S^R by S and R respectively. We conservatively assume that the clock drift is lower than $0.05 \cdot T$; Marouani et al., report that a clock variation of 0.5 ppm, i.e., a drift of 0.5 μs per second, is a conservative estimate [123].

Furthermore, to account for the worst case, we assume that R has the fastest clock, i.e., R thinks T has passed when in reality $0.95 \cdot T$ has passed, and S has the slowest clock, i.e., S thinks T has passed when in reality $1.05 \cdot T$ has passed. Thus, we set $\Delta = 0.1 \cdot T$.

Sequence-number window length (M). Equation 4.1 expresses the length of the seqNo window that a router maintains, as a function of the period that a single seqNo is used (T) and of the maximum latency variance (σ) between two packets.

The inequality is derived as follows: let t_i denote the time at which S starts using seqNo i. Since seqNos are updated every T, the elapsed time between the start of two seqNos is $t_i - t_j = (i-j) \cdot T$, for $i > j$. Then, consider a router that accepts packets in the window $[i, i+M-1]$. To ensure that a legitimate packet is not dropped due to packet reordering, the last packet with a seqNo of i should arrive at R before the first packet with a seqNo of $i+M$ (Figure 4.5). In the worst case, the last packet with seqNo i is sent at $t_{i+1} - \epsilon_1$ and received at $t_{i+1} - \epsilon_1 + \sigma$, where ϵ_1 is an arbitrarily small positive constant. The first packet with seqNo $i+M$ arrives at R no earlier than $t_{i+M} + \epsilon_2$, where ϵ_2 is an arbitrarily small positive constant. Mathematically, the relation between the packet-arrival times can be described as $t_{i+1} - \epsilon_1 + \sigma < t_{i+M} + \epsilon_2$. By rearranging the inequality and using the fact $t_{i+M} - t_{i+1} = (M-1) \cdot T$, we obtain Equation 4.1.

Bloom-filter parameters (m, k, N, L). Equation 4.2 expresses the number N of required BFs as a function of the BF rotation interval L. Recall that

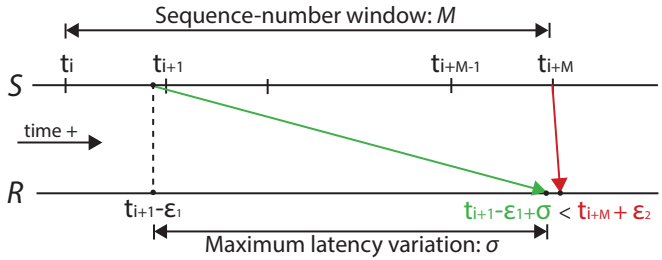


Figure 4.5: The last arriving packet with seqNo i must arrive before the first arriving packet with seqNo $i+M$.

R has to store a packet at least until the seqNo of the packet is no longer valid, which is a time period of length $M \cdot (T+\Delta)$.[4] Therefore the complete rotation of the circular BFs, which lasts for $N \cdot L$, should take longer than $M \cdot (T+\Delta)$, and this yields that $N \cdot L > M \cdot (T+\Delta)$. We obtain that $N > M \cdot (T+\Delta)/L + 1$ filters are required; the additional filter is necessary to store the incoming packets at the current time interval. We combine the formed inequality with Equation 4.1 and by setting $\Delta = 0.1 \cdot T$ we obtain Equation 4.2.

Equation 4.3 describes the size m of each BF as a function of the BF rotation interval L, the number N of BFs, the number k of necessary hash functions, and the BF's target false-positive rate (fp). Since an incoming packet is checked against all BFs, the overall target false-positive rate is $1-(1-fp)^N$. To determine the value for fp, we consider the average number of packets that a router receives in an interval L (which is $r \cdot L$, where r is the incoming packet rate). Using the BF equations, we get $fp = (1 - e^{k \cdot x \cdot L/m})^k$ and by combining it with the equation for the size of a BF, we obtain Equation 4.3. The inequality indicates that any larger value for m yields a lower false-positive than fp.

The formulated optimization problem is an integer programming problem, which is know to be NP-hard [138]. Note that also the rotation interval L is an integer: a time period in a computing system is expressed as a multiple of some minimum supported time granularity (e.g., 1 ns); in practice, we will use values at the order of 1 ms (Section 4.4). In our

[4] In the worst case, a router does not receive seqNo updates from the source and self-increments the seqNo every $T + \Delta$.

context, we obtain multiple solutions to the problem by searching a constrained parameter space; for example, we constrain the size of the BF to be less than 20 MB, since ideally it should fit into a processor's cache. Our grid search is performed as follows: $10\ ms \leq L \leq 200\ ms$, $2 \leq N \leq 20$, $2 \leq k \leq 30$, $1\ MB \leq m \leq 20\ MB$.

Furthermore, the objective of the optimization problem changes depending on the implementation platform (e.g., software vs. hardware-based implementation). In Section 4.4.1, we adapt the optimization problem to a software router and show how a selection of carefully chosen parameters leads to an efficient implementation.

4.4 Software Prototype

To demonstrate the practicality of our approach, we implement the proposed replay-suppression mechanism on a software router. Our evaluation focuses on the overhead of replay suppression and not other functionalities (e.g., source authentication or longest prefix matching). Thus, we do not consider a specific underlying network architecture, but we make the following generic assumptions:

- Every packet injected into the network by a host has a unique network-layer identifier. For example, the IP-ID field in IPv4 is implemented by most operating systems as a packet counter [165]. We use this identifier together with the immutable content of a packet to uniquely identify the packet and minimize the probability of a collision.
- A router can obtain the AS number (ASN) of the source-host for every packet. For example, certain network architectures express addresses as a $(ASN : hostID)$ tuple [25, 117]; or an IP forwarding information base (FIB) can be extended to include this information for every source-address prefix.

4.4.1 Implementation

The main focus of our implementation is to optimize memory-access patterns. Since our solution is a memory-intensive application,[5] forwarding performance depends mostly on cache efficiency, i.e., it depends on the

[5] For each packet, $k \cdot N$ bits are accessed in the BFs; k bits for each one of the N BFs.

Parameter	Value	Parameter	Value
T	10 ms	m	8 MB
r	14.88 Mpps	k	11
σ	100 ms	N	2
M	11	L	121 ms

Table 4.2: Software-router implementation.

memory footprint of the application and on the memory access patterns. Small data structures are more likely to fit in the cache and, thus, reduce the importance of the access pattern. However, in a software implementation the cache is shared with other processes and a small memory footprint does not guarantee optimal performance. Thus, we focus on minimizing cache misses.

To minimize cache misses, we use a blocked BF [70] instead of a standard BF. A blocked BF consists of multiple standard BFs (called blocks), each of which fits into the typical 64-byte cache line. For each element that is checked/inserted, the first hash value determines the block to be used and additional hash values determine which bits to check/set in the block. Thus, a blocked BF incurs only one cache miss for every operation in the worst case. This optimization comes at the cost of a larger memory footprint compared to a standard BF with the same false-positive rate.

The next step to minimize cache misses is to minimize the number of blocked BFs. Recall from the protocol description (Section 4.3.2) that for every observed packet, we add it to the writeable BF and check for its presence in all other filters. Since blocked BFs may have one cache miss per checked/inserted element, we want to minimize the number of filters.

We solve the optimization problem (Section 4.3.3) with the objective of minimizing the number of BFs. To account for the worst case, we assume a packet rate of 14.88 Mpps, which is the theoretically maximum packet rate for a 10 GbE Network Interface Card. Also, we set a conservative value for the maximum latency variation σ to 100 ms, based on a recent latency-measurement study [60]. We target for an overall false-positive rate that is less than $5 \cdot 10^{-6}$, and we obtain multiple solutions that use $N = 2$ BFs (which is also the lowest possible value according to Equation 4.2). Specifically, we obtain solutions that have different filter sizes (m), different seqNo window lengths (M), and that rotate BFs at different

time intervals (L). From these solutions we choose the one that has the smallest memory footprint (lowest m value), under the constraint that the filter size is a power of 2. This constraint provides a significant processing speedup, as heavily used computations are transformed to bitwise operations (e.g., modulo operations become bit-shifts). Table 4.2 summarizes all the parameters of our solution.

Furthermore, to check/insert elements in the BF, we need to obtain the pointers to the corresponding bits in the filter. To implement the keyed PRF (Section 4.3.2), we compute an AES based CBC-MAC over a *fixed length* of the first bytes of a packet, as a CBC-MAC is insecure for variable-length messages [34]. Also, from our analysis of CAIDA traces [7], we found that the first 48 bytes of a packet's content are sufficient to mitigate digest collisions; the same result has been reported by previous work [58]. We split the 16-byte output of the MAC into appropriately sized chunks so that the first chunk points to the 512-bit block and the remaining chunks point to the bits in the block.

The last required functionality is the FIB. The FIB holds for every AS S the seqNo SN_S^R and the count-down timer TTL_S; we decrement the TTL value every 1 ms.

Optimizations. We leverage the Data Plane Development Kit (DPDK) [13] and Intel AES-NI [76] to build our prototype, and we perform the following optimizations to the BF. To insert an element, we leverage 128-bit registers and an SSE *OR* instruction: we prepare the inserting element by setting the respective bits obtained from the MAC computation. Then, we set the required bits in the 512-bit block with four 128-bit SSE OR operations. To check for membership of an element, we use early exit, i.e., as soon as we discover an unset bit we know the element is not a duplicate. This results in better performance since false positives are low and it is very likely to discover unset bits early.

4.4.2 Evaluation

We evaluate the switching performance of our software router on a commodity server equipped with an Intel Xeon E5-2680 CPU (20 MB L3 cache), 32 GB DDR3 RAM, and a 10GbE Network Interface Card (NIC). We dedicate only two cores of the CPU to perform all required processing: one

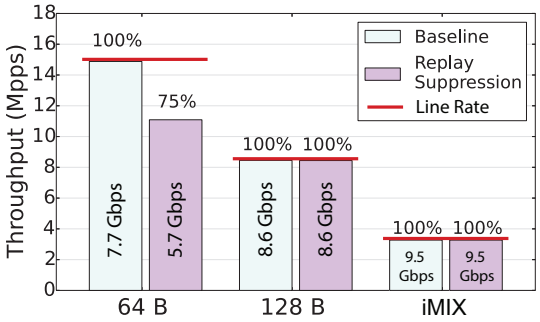


Figure 4.6: Forwarding performance for packet sizes of 64 and 128 bytes and for IMIX.

core handles all the packet processing, and the other core updates the TTL values and seqNos in the FIB.

We utilize Spirent SPR-N4U-220 as our packet generator to generate load on the router; the router processes the generated traffic and sends it back to the generator. We generate a FIB with 55k ASNs, and use random destination addresses to avoid spatiotemporal locality for FIB cache accesses.

First, we test the forwarding performance of one port for two packet sizes (64 and 128 bytes) and a representative mixture of Internet packet sizes (IMIX) [129]. Minimum-sized packets, 64 bytes, translate to the highest possible packet rate and are the worst case for the software router; we refer to the highest packet rate for each test case as the line-rate performance. The baseline for the experiments is the forwarding performance without any packet processing. Figure 4.6 shows the forwarding performance we obtain. The results show a 25% decrease for minimum-sized packets; and that for longer packet sizes, i.e., lower packet rates, optimal performance is achieved.

Next, we measure the latency overhead of our implementation (Figure 4.7). We observe a two-fold increase in average latency only for minimum-sized packets. The average latency and latency range is almost identical for the other two test cases.

We observe a performance degradation, both for throughput and latency, for minimum-sized packets. This performance degradation is attributed to the penalty of the cache misses and the overhead of the MAC

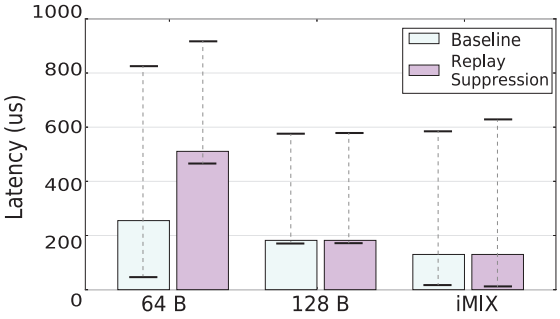


Figure 4.7: Average, minimum, and maximum packet latencies for packet sizes of 64 and 128 bytes and for IMIX.

computation when the router is subjected to the maximum load. We emphasize that a 10 GbE link, fully utilized with 64-byte packets is far from any realistic workload. IMIX is a representative of common Internet traffic and has an average packet size of 417 bytes; our implementation saturates line-rate for IMIX and 128-byte packets.

4.5 Security Considerations

4.5.1 Deployment Location and Topology

We discuss certain security issues that depend on the location of routers that deploy replay suppression and on the network topology.

In Section 4.3, we mentioned that routers which deploy replay suppression are located at the borders of ASes. This deployment model raises certain security issues, which are mitigated if more routers inside an AS deploy the protocol.

Packet replays create an attack surface, which includes the path segments between the malicious router and the first deploying router that will drop the replayed packets. If deploying routers are located only at AS borders, then the attack surface is limited to a single AS. If more routers inside an AS deploy the protocol, then the attack surface is further reduced. For example, ASes could deploy more such routers near routing bottlenecks.

Furthermore, a malicious router can strategically replay packets even against a deploying routers: since replay suppression is done by routers

individually, without coordination among them, a packet that is sent to one deploying router can be successfully replayed (once) to another deploying router. However, the effect of such an attack is limited: the malicious router can replay a packet at most once to a router that performs replay suppression; additional replays after the first one will be suppressed.

4.5.2 Attacks on Bloom Filters

BF implementations are a common target for attackers [74]. We consider two types of attacks that could be launched against our protocol.

1. **Chosen-insertion Attack.** In a chosen-insertion attack, the adversary crafts packets that fill up the bits in the BF so that the false-positive rate becomes very high. Our protocol is resilient against the attack because we use a keyed PRF to compute the bit locations in the filter. Since the key is not known to the adversary and the output of the PRF is uniformly random, the adversary cannot set specific bits in the filter.

2. **Query-only Attack.** In the query-only attack, the adversary attempts to launch a DoS attack against the BF by querying items that take an abnormally long time to check. In our protocol, we focus on cache efficiency, so that either the BFs fit entirely in the cache or that checking for an item requires at most one cache miss.

4.5.3 Sequence-number Wrap-around

The use of per-interval seqNos makes wrap-arounds infrequent,[6] but wrap-arounds, i.e., restarting from zero, will eventually happen. In the event of a wrap-around, previously invalidated packets can be replayed because the seqNos become valid again. Furthermore, an adversary can replay previously seen packets that have higher seqNos than the one currently used by a source AS, so that the router would fast-forward the seqNo window for the source AS. As a result, the router would drop all legitimate packets that are sent by the source AS.

[6] For a 4-byte seqNo that is incremented every 10 ms, it takes about 497 days to wrap-around.

This problem is inherently solved by the underlying source authentication mechanism. In source-authentication schemes, source ASes periodically update their keys, so if the source AS updates the key before its seqNo wraps around, then the old packets will be invalid due to an authentication failure.

4.6 Discussion

Hardware Implementation. In our software-router implementation, we used blocked BFs because the cache is shared with other processes.

In a hardware implementation, however, the optimization objective changes since a NIC can have a dedicated cache for the purpose of replay suppression. Standard bloom filters are a better option, because they are more space efficient than blocked bloom filters (for a given false-positive rate) and thus, can potentially fit into a dedicated cache. Table 4.3 summarizes the parameters of the optimization problem for a hardware-based implementation; the aggregate footprint of the application is less than 12 MB, with a false-positive rate of $9.85 \cdot 10^{-7}$.

Compliance to Sequence-number-update Interval. Recall that the only parameter that requires global agreement is the interval T at which ASes update their sequence numbers. We stress that ASes have no incentive to deviate from T. If AS S updates SN_S too fast, S may experience packet dropping due to packet reordering: packets with higher sequence numbers may arrive faster than packets with lower sequence numbers; this risk increases as T becomes smaller. If S updates SN_S too slowly, S may experience packet dropping due to low seqNos: a router R may self-update its seqNo for S and the seqNos in the packets may fall out of the seqNo window.

Failure Recovery. Intermediate routers maintain seqNos and previously forwarded packets for all ASes. In an event of a failure (e.g., loss of power), a router may lose this information and thus, is unable to identify replayed packets after reboot. This is only a temporary situation: upon receiving a packet from a source AS, the router synchronizes its seqNo for the source, allowing it to filter any packets that fall out of the seqNo window, but not replayed packets with valid seqNos; however, after at most $M \cdot (T + \Delta)$ since the seqNo update, the router has fully recovered from the failure and can suppress all replayed packets.

Parameter	Value	Parameter	Value
T	10 ms	m	4 MB
r	14.88 Mpps	k	11
σ	100 ms	N	3
M	11	L	61 ms

Table 4.3: Hardware-based implementation.

Routers of the source AS may fail as well. In this case, the router obtains the seqNo that the AS is using and the time for the next seqNo update by asking neighboring routers within the same AS.

A source AS (or a router) may fail as well. In case of a router failure in the source AS, the router asks a neighboring router within the same AS to determine the current seqNo as well as the time for next seqNo update. In case of a catastrophic failure of the entire AS, there are three choices. The source AS could use a sufficiently high seqNo so that routers in other ASes can synchronize to the new seqNo of the source AS. Or, the source AS can ask its neighboring ASes (or their neighboring routers while re-establishing BGP sessions) for the seqNo that it was using prior to failure. Alternately, as a last resort, a transit AS can erase state for the source AS, if it does not observe traffic from the source AS for a sufficiently large period of time; then, the transit AS uses the seqNo of the source AS when it observes again traffic from the source AS.

Chapter 5
Deployment Strategies

Deploying a future Internet architecture (FIA) that changes the core of the Internet is an onerous task. To make matters worse, there is no systematic formula or guideline to successfully deploy such architecture; only heuristics and recommendations exist based on previous experiences.

A well-known approach to deploy a new architecture is by constructing an overlay network of the architecture over today's Internet. That is, today's Internet is used as *virtual links* to interconnect components of the new architecture that are geographically separated. Then over time, as the demand for the new network technology becomes ubiquitous and pressing, the new architecture gradually assumes the role of today's Internet. The most successful example of this deployment path is today's Internet which was initially deployed as an overlay over the public telephone network [52]. Moreover, the IPv6 deployment is following this deployment path: in its early stage, IPv6 networks were deployed as overlay networks over today's Internet. As the demand for IPv6 increases (primarily due to the imminent depletion of IPv4 addresses), the IPv6 network is gradually expanding to become the network architecture for today's Internet [164].

However, there is no conclusive evidence that the overlay approach is the most appropriate approach for FIA deployment. Moreover, given the diversity of FIA proposals, it is unlikely that the overlay approach would be the best approach for all FIA proposals. Hence, in this chapter, we explore an alternate approach for FIA deployment in addition to the overlay approach.

For the alternate approach, we start by examining the goal of the FIA research. The ultimate goal of the FIA research is to design and deploy "a network that serves all the needs of society" [136]; however, there are different views on how FIA research efforts work towards this goal [72, 152]. In one view, the goal of the FIA research is to design an architecture that will replace or coexist with today's Internet; and, this goal is reflected on the overlay deployment approach. In another view, the goal is to gain a systematic understanding of the network to establish a *networking disci-*

pline. Yet from a different angle, the goal is to identify new ideas that were previously not thought of and realize those ideas on today's Internet.

We take on the last view on the goal of the FIA research and propose a deployment approach that can be summarized by the following phrase—*think revolutionary but act evolutionary*. In this approach, the FIA research serves as a platform to think revolutionary without confining oneself to the restrictions and the limitations of today's Internet to identify new ideas. Then, these ideas are transformed and incrementally adopted on today's Internet. In this approach, a deployment is successful if the ideas of a FIA are realized on today's Internet. We name this alternate approach as the *integrated approach* to emphasize that the main ideas of the deploying FIA are integrated on today's Internet instead of creating an overlay network for the FIA.

We apply the integrated approach to our architecture, APNA. To this end, we take one idea of APNA—user-defined privacy—and explore how that idea can be realized on today's Internet using practical and readily deployable technologies.

To this end, we take a user-centric approach to privacy, and our first step is to account for users' diverse privacy requirements. Following a definition of privacy by Westin [172]—what is revealed to whom—we introduce the term of *privacy domains*. A privacy domain is defined by the entities (e.g., ISPs) and the privacy-sensitive information (e.g., source address) that is revealed to these entities for a user's communication session. Privacy domains help us in bridging the gap between users' high-level privacy requirements and the more actionable technical requirements.

Our second step is to identify simple and common networking practices that can be used to realize privacy domains. We identify three such practices—encryption, address translation, and tunneling—that can be combined or used in isolation to construct privacy domains.

Then, based on these common networking practices, we propose three privacy services that can be offered by ISPs: i) An address-hiding service that enables customers to use a different IP address for every traffic flow. ii) An ISP-wide tunneling service so that tunneled traffic between source and destination ISPs is encrypted. iii) An ISP-level VPN service that remote hosts of other ISPs can use.

We argue that ISPs are in an ideal market position to offer such privacy services. They already have high-capacity infrastructure in place, which they can use to offer services at a large scale. Furthermore, they have the required know-how and experience in deploying and operating large systems. Our initial evaluation results indicate that the services can be offered at a low overhead, even on today's commodity hardware. In addition, our proposed services have a low deployment barrier, offering incentives for first movers.

In this chapter, we make contributions in three directions:

- We introduce a concept—privacy domains—to express users' privacy requirements at a high-level, irrespective of the underlying implementation.
- We describe common networking practices and techniques that can serve as building blocks to implement privacy domains.
- We present simple privacy services (with a preliminary feasibility analysis) that can be offered by ISPs.

We explore privacy from a new perspective: leveraging existing technologies to enable ISP-based privacy services, which can fill the large trade-off space between privacy and performance.

5.1 Overlay Approach

In this section, we describe the overlay approach to deploy APNA.

5.1.1 APNA Overlay Network

Our ideas are not restricted to a certain Internet architecture, thus APNA could be used on today's Internet. To this end, IPv4 addresses of the hosts are mapped to HIDs using a mapping table, and IPv4 addresses of APNA routers in ASes serve as AIDs. Furthermore, we leverage the GRE protocol [64] 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 as shown in Figure 5.1; 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.

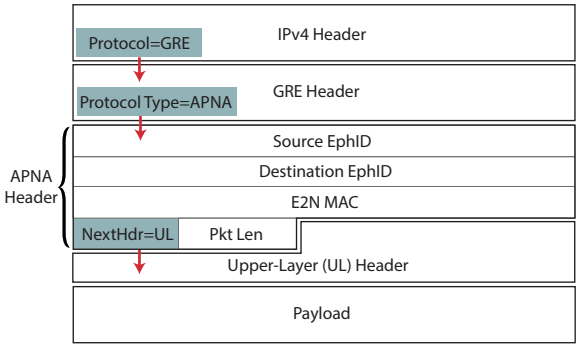


Figure 5.1: IP+APNA packet structure.

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 of the destination host; and, 2) translates the HID to the corresponding IPv4 address; and 3) replaces the destination IPv4 address of the IPv4 header with the translated IPv4 address.

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 (i.e., IPv4 address of an APNA router at the destination AS) 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.

5.1.2 APNA Gateway

Making modifications to the host network stack is an onerous task that hampers deployment of new architectures. Hence, we propose using APNA

gateways to bridge between the Internet and APNA without having to change the host network stack. An APNA gateway has two roles: 1) as an APNA host, it runs the protocols described in Section 2.3; and 2) as a packet translator, it converts between native IPv4 and APNA packets. Assuming that the gateway uses different source EphIDs for different IPv4 flows, the challenge in translating between IPv4 and APNA packets is determining the mapping between IPv4 flow information (identified by the standard 5-tuple) in IPv4 packets and APNA flow information (identified by source and destination AID:EphID pair) in APNA packets.

When forwarding an outgoing IPv4 packet from a host to an APNA router, the gateway converts the IPv4 packet to a APNA packet (Figure 5.1). To this end, the gateway must determine source and destination AID:EphID tuples for the APNA header, and source and destination IPv4 addresses for the IPv4 encapsulation header. Determining the addresses for the encapsulation header is simple—the addresses of the gateway and the APNA router are used as the source and destination IP addresses, respectively. In addition, the source AID:EphID information in the APNA header can be easily determined: for each new IPv4 flow, the gateway uses a different EphID. However, determining the destination AID:EphID is not trivial. In fact, the gateway cannot determine the destination AID:EphID solely based on the information in the IPv4 packet from the host.

Instead, the gateway has to rely on mechanisms that the host uses to determine the address of its peer host. For instance, if a client uses DNS to resolve the IPv4 address of the server, we can extend the DNS record to also contain the AID:EphID of the server and include the AID:EphID in the DNS reply message. Then, from the DNS reply message, the gateway associates server's IPv4 address to the corresponding AID:EphID. Note that this learning approach can only work if a host use a well-known mechanisms (e.g., DNS) to determine his destinations' addresses. Otherwise, the host needs to statically configure the mapping between a peer's IPv4 address and AID:EphID information into the gateway.

In the above client-server communication example, one may argue that the host privacy of the server is lost since its IPv4 address is registered in DNS. To overcome such privacy loss, the IPv4 address can be removed from the DNS record. When the client's gateway sees the DNS reply, it generates and appends a random IPv4 address into the DNS reply. Then,

based on the destination IPv4 address in the client's packets to the server, the gateway determines the AID:EphID of the server.

When forwarding an incoming APNA packet from an APNA router to a host, the gateway needs to convert it to an IPv4 packet by choosing appropriate source and destination IPv4 addresses. If the gateway already has the mapping between the APNA flow tuple and the IPv4 flow tuple (i.e., the receiving host has sent an outgoing packet with the IPv4 flow tuple), the gateway uses the IPv4 flow tuple to create the IPv4 packet. However, if the gateway does not have the mapping, the gateway needs to carefully choose the source and destination IPv4 addresses for the IPv4 packet. We first describe how we determine the source addresses, then the destination addresses.

When choosing the source address, the gateway needs to ensure that the host can distinguish between different flows. That is, every APNA flow tuple must be mapped to a unique IPv4 flow tuple. To this end, we define a *virtual end-point* which consists of an IPv4 address (e.g., randomly drawn from a private address space), and the source port number in the transport header in the APNA packet. The gateway assigns unique virtual end-point for each APNA flow, and the IPv4 address of the virtual end-point is used as the source IPv4 address in the IPv4 packet.

To determine the destination IPv4 address, the gateway uses the destination EphID information in the APNA header. However, the mapping between EphID and IPv4 address exists only if the destination has sent an outgoing packet or the destination host has registered the mapping between its EphID and IPv4 address. For example, a server administrator registers a (receive-only EphID, IP address)-tuple to his gateway after registering his domain information in DNS.

5.2 Integrated Approach

In this section, we apply the integrated approach to deploy APNA.

5.2.1 Privacy Domains

We observe that users have diverse privacy requirements, yet few options or tools to use. Our starting point is to introduce the concept of privacy domains, which enable us to argue about disclosed information at a higher layer of abstraction without considering a specific privacy architecture nor

a specific adversary model; existing work focuses on architecture-specific analysis that consider specific threat models [55, 160].

For example, a common privacy requirement is to prevent entities that observe traffic from creating a history of user activity (e.g., list of hosts the user communicated with). This high-level requirement is translated to the requirement that no entity can observe the source and destination addresses of a communication session at the same time. To facilitate the translation, we define *privacy domains*.

A *privacy domain* is a virtual domain that consists of entities to which a user shares a subset of her privacy-sensitive information. Thus, a privacy domain is defined by a set of a user's privacy-sensitive information and the entities that have access to this information.

- **Entities:** We consider entities in a privacy domain from a sender's view-point, i.e., entities that assist in transferring packets from the sender to the destination. Such entities include the ISP of the sender, the transit ISPs on the path to the destination, the destination ISP, and the destination host.
- **Privacy-sensitive Information.** We consider information that is revealed about the sender through sent packets. Specifically, we consider the source and destination host addresses, the source and destination ISPs, the transport-layer headers, and the plaintext payloads.

Although the notion of privacy domains has not been used so far, there are various implicit privacy domains for Internet communication. They are created when a user leverages privacy-enhancing technologies. We describe the privacy domains for two commonly used technologies:

- **TLS protocol:** When a client connects to a server over TLS, two privacy domains are created. Domain *A* consists of the server with whom the client shares all its privacy-sensitive information. Domain *B* consists of all other entities that observe traffic, i.e., all ISPs on the path. The client shares all privacy-sensitive information with Domain *B*, except for the payload since it is encrypted.
- **Tor:** If the client connects to the server using Tor [56], three privacy domains are created, assuming that the client uses a typical three-hop Tor Circuit with an entry, a transit, and an exit relays. Domain

A consists of the Tor entry relay, the entry relay's ISP, the client's ISP, and all transit ISPs between the two ISPs. The client shares only her address with domain *A*. Domain *B* consists of the Tor exit relay and its ISP, the destination host and its ISP, and all transit ISPs between the two ISPs. The client shares with domain *B* the address of the destination host, the transport-layer header and the payload. Note that the combination of source and destination addresses is not present in any of the two domains. This property enables a user to hide her history of user activity from all entities—at least in theory. Lastly, Domain *C* consists of the Tor relays (i.e., entry, transit, and exit) relays, their ISPs, and the transit ISPs between the relays. The client does not share any privacy-sensitive information in this domain.

By defining privacy domains, our goal is to clarify and articulate information disclosure, not to evaluate the privacy guarantees of a certain mechanism. For example, when a user uses Tor, the combination of source and destination addresses is not present in any of the Domains (i.e., Domains *A*, *B*, and *C*); however, there are sophisticated attacks to infer the combination [162, 168, 169, 171, 179]. Existing approaches can be used to evaluate the privacy guarantees offered by certain mechanisms [126].

5.2.2 Overview

We start with basic building blocks that can help in constructing privacy domains. Then, we present three privacy services that are based on these blocks and can be offered by ISPs.

5.2.2.1 Building Blocks

Encryption masks privacy-sensitive information from unwanted parties. In its most common use case—the TLS protocol—encryption masks the payload so that only the destination can see it in plaintext. Alternately, encryption can also mask the transport-layer headers, thus hiding the application being used, as happens in IPsec transport mode.

As a generic concept, encryption creates two privacy domains: one domain that is defined by the entities that have the decryption keys and therefore have access to all privacy-sensitive information; and the other domain that consists of all remaining entities that have access only to the unencrypted information in the packets.

Address Translation is used so that multiple users can share the scarce IPv4 address space. A side-effect of address translation is that an observer located after the translation point cannot identify the original source host of a packet.

Address translation creates two privacy domains: one domain that consists of the entities that know the original source address of the sender (and all other privacy-sensitive information), and one domain that consists of the entities that see all information except for the original source address.

Source addresses in the packets are translated to another addresses so that an unwanted party cannot identify the original source addresses. This basic building block is useful for hosts whose addresses can be (easily) related to their identities.

Address Translation creates one privacy domains consisting of entities that know the original source address of the sender.

Tunneling bridges two networks by creating a virtual point-to-point link; it is typically used to provision a service that the underlying network cannot support (e.g., supporting IPv6 over the IPv4 network). Tunneling by itself does not define new privacy domains, but can be used for this purpose. More specifically, it enables the sender to specify a waypoint that the traffic should follow towards the destination. Then, with the help of the waypoint and in combination with the previously mentioned building blocks new privacy domains can be defined (See the ISP-wide Secure Tunneling Service and the ISP-level VPN Service in Section 5.2.2.2).

5.2.2.2 ISP-based Privacy Services

We describe privacy services that ISPs can offer; the services are based on the building blocks described, and thus compatible with today's networking practices.

ISP-level Address Hiding Service (AHS). We start with an address-hiding service (AHS) that is based on *source-address translation* and can be offered by ISPs to its hosts. When subscribers' packets exit the ISP boundary, the source addresses are replaced with other addresses in the ISP's address pool. Specifically, the source address is replaced with a different address for every outgoing flow. When subscribers' packets enter the ISP's boundary, the reverse translation takes place so that the packets are forwarded to the intended recipients. Hosts remain agnostic of AHS in that they pur-

chase the service but they do not need to upgrade the OS nor run specialized applications.

We are not the first to propose address shuffling by ISPs. In 2009, Raghavan et al. [149] proposed the use of a tweakable block cipher to enable ISPs to shuffle their IP addresses; we build on the same motivation, but construct mechanisms that provide a higher degree of flexibility to ISPs (Section 5.2.3.1).

Privacy Domains. The AHS creates two privacy domains for a subscribing host. One domain consists of the host's ISP, and the host shares all its information with this domain. The other domain consists of all other entities; the host shares all information except for the source address, which is instead shared at the granularity of an AS.[1]

Use Cases. This service is useful for users who want to hide their online activity, i.e., which hosts they contact; the online activity is still disclosed to the user's ISP.

ISP-wide Secure Tunneling Service (STS). We propose ISPs to offer a secure tunneling service (STS) by setting up encrypted tunnels with other ISPs; source and destination ASes are the tunnel end points. The traffic is encrypted by the source AS when it enters the tunnel and then decrypted by the destination AS when it exits the tunnel. Similar to AHS, hosts remain agnostic of the tunneling service and do not need to upgrade or run additional software.

Privacy Domains. This service is composed of two basic building blocks: tunneling and encryption. Tunneling specifies two waypoints on the path to the destination, and the waypoints encrypt/decrypt the traffic, hiding it from other entities on the path.

STS creates two privacy domains for a subscribing host. One privacy domain consists of the source ISP, the destination ISP, and the destination host; the subscriber shares all its information with this domain. The other privacy domain consists of all transit ISPs, and the subscriber shares its address and the address of the destination host at the AS granularity.

Use Cases. ISP-wide tunnels provide an additional security measure for traffic that is already encrypted. Services that exchange sensitive information typically perform encryption with TLS at the application layer, though

[1] We use the term "ISP" when referring to services, and the term "AS" mostly for protocol-level details.

this is not always sufficient. Protocol and implementation vulnerabilities of popular TLS libraries have enabled decryptions at a large scale (e.g., the Heartbleed attack [11] and compression attacks [8, 9, 10]). Moreover, the lack of forward secrecy in some TLS deployments can lead to compromised plaintexts, if a long-term key is compromised—about 30% of 200K popular TLS-enabled websites still do not fully support forward secrecy [24].

In addition, ISP-wide tunnels can provide a layer of security for unencrypted webpages. Today's trend moves towards pervasive encryption, and although TLS is gaining traction, we are far from universally encrypted traffic—less than 20% of the top 10k websites and less than 0.1% of all websites have TLS enabled by default [23].

ISP-wide tunnels can also harden today's privacy protocols. Tor is known to be vulnerable against traffic correlation attacks when an adversary can observe traffic at the entry and exit points of the Tor network [94]. Even worse, an adversary can launch BGP prefix hijacking attacks to position itself on the path of inbound and outbound traffic [168].

STS alleviates the consequences of today's insecure inter-domain routing. More precisely, adversaries often launch BGP prefix hijacking attacks to attract traffic [19, 20, 21, 22]. They can then analyze traffic patterns (who talks with whom), modify and inspect unencrypted traffic, or store encrypted traffic with the prospect of breaking it in the future. Using ISP-wide tunnels does not prevent BGP hijacks, but limits the capabilities of attackers through strong network-layer encryption for all traffic in the tunnel.

ISP-level VPN Service (IVS) We propose ISPs to offer VPN services to hosts of other ISPs, similar to the VPN services that already exist on the market. The motivation for this proposal is that despite the increased interest in VPN services, the market is littered with low quality services. Reasonable performance is offered only by premium services that cost around $10 per month—a considerable fraction of an Internet connection's monthly cost.[2] Furthermore, most such services suffer from critical vulnerabilities (e.g., IPv6 traffic leakage and DNS hijacking) that disclose the identity and traffic payloads of users [142].

We argue that ISPs are in a better market position to offer VPN services at a fraction of what VPN services cost today. ISPs have high-capacity in-

[2] http://to.pbs.org/1z5AGNQ

frastructure and experience in deploying and operating services at a large scale. Furthermore, ISPs already manage large blocks of IP addresses, which are necessary to mask customer identities through address translation; it is commonly the case that today's VPN services point customers to connect to servers at a different location due to their IP address scarcity at some locations. Thus, we believe ISPs can leverage their experience and economies of scale to gain an extra source of revenue.

We propose that ISPs offer VPN services to which hosts of other ISPs can connect. This service is based on all three basic building blocks: combining *encryption* and *tunneling* to create an encrypted tunnel between a subscriber and an ISP that offers the VPN service, and *address translation* to replace the source address in a subscriber's packet with an address of VS offering ISP.

Privacy Domains. This service combines all three building blocks: tunneling, encryption, and address translation. Tunneling is used to specify the VPN provider as a waypoint for the traffic towards the destination host, and encryption hides the traffic from entities on the path to the VPN provider. Then, address translation replaces the subscriber's source address with another address in the provider's address pool.

Three privacy domains are created for a subscribing host. One domain consists of the ISP offering VPN service, and the subscriber shares all its information with this domain. Another domain contains the source ISP and all transit ISPs leading to the VPN provider, and the subscriber shares only its source address with this domain. The last domain contains all transit ISPs between the VPN provider and the destination ISP, the destination ISP, and the destination host; the subscriber shares all information except for its source address.

Use Cases. Users may use VPN services for different purposes: i) to bypass geolocation restrictions, ii) to circumvent governmental censorship, and iii) to hide their activity from their ISPs by encrypting traffic and hiding the destination.

5.2.3 Feasibility Study

Our goal is to provide deployable privacy services for today's Internet, and therefore we build on well-established practices. Although this ensures interoperability with today's protocols, there are open problems when dealing with thousands of inter-connected networks and large ISP deployments.

We address such challenges and provide preliminary evaluation to assess the feasibility of our proposal.

5.2.3.1 ISP-level Address Hiding

We propose ISPs to shuffle IP addresses of hosts subscribed to the AHS. When the ISP receives an outgoing packet, a translation gateway creates a mapping from the packet's source IP and port number to a new IP address and port number. This new IP address belongs to one of the ISP's address blocks; the ISP can use the bits in the IP's host portion together with the bits of the source port in order to multiplex different flows behind a few IP addresses. When the ISP receives an incoming packet, the gateway performs the reverse translation so as not to break bidirectional communication. This process may remind of a carrier-grade NAT [92, 175], yet there are multiple challenges that we address in the following.

Coordination of Translation Gateways. An ISP will operate multiple translation gateways and they must all perform identical translations. First, translation must be performed on the original data path to minimize latency overhead; rerouting traffic to a centralized location would cause a latency inflation. Therefore, an ISP will operate multiple gateways, but under the constraint of identical translations. This is to prevent collisions so that two different inputs do not generate the same output. Also, the translation of an outgoing packet and the reverse translation for the incoming packet may be performed by different gateways due to asymmetric routing.

One naive approach to achieve identical translations is to exchange mapping tables between gateways. However, this solution is not viable since such mappings must be distributed for every new flow and to all gateways. We leverage cryptography to perform the translation without keeping per-flow state [149]: translation gateways encrypt the source address and port tuple and generate a new tuple, under the constraint that the network prefix belongs to the ISP. All gateways share the same key so that all (reverse) translations are identical; the state stored at every gateway is just the encryption key. This approach satisfies all the constraints mentioned.

Privacy vs. Traffic Engineering. The privacy benefits of the AHS service come from the fact that ISPs have large IP blocks, which provide a satisfying anonymity set when permuting hosts' addresses. However, ISPs

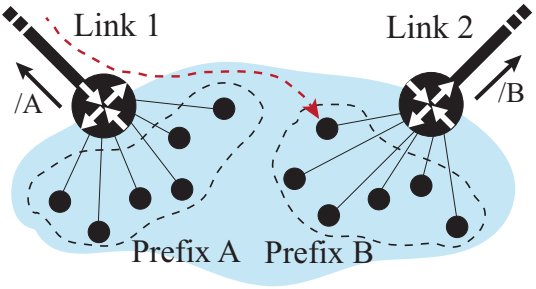


Figure 5.2: Example with an ISP owning two IP blocks and using two inter-domain links.

deaggregate these address blocks in their BGP announcements in their effort to perform fine-granular traffic engineering. Although this practice still allows address shuffling, it reduces the anonymity set to the size of an advertised block.

We illustrate with an example in Figure 5.2. The ISP owns two prefixes and is connected to the rest of the Internet through two inter-domain links. The addresses of Prefix A are assigned to the marked hosts on the left side, and Prefix A is announced over Link 1. Similarly, Prefix B, assigned to the hosts on the right side, is announced over Link 2. This represents a simplified, but common scenario of prefix deaggregation and traffic engineering.

Assume the ISP starts offering the AHS service and has two translation gateways collocated with the two border routers. From a privacy perspective, the ISP ideally should shuffle addresses from both prefixes in order to maximize the anonymity set. This means that e.g., hosts on the right side may receive an address from Prefix A (even if the traffic is going out from Prefix B). However, the corresponding incoming packet would enter the ISP from Link 1 and follow a sub-optimal route to the intended host (dashed arrow). Thus, shuffling larger block sizes constraints the granularity at which traffic engineering can be performed.

We cannot eliminate this conflict, but we must provide flexibility to ISPs in picking the desired tradeoff. Therefore, we design a translation mechanism that enables shuffling for prefix blocks of arbitrary size. This task raises a challenge, since most cryptographic primitives operate on input of fixed length, e.g., 128-bit block for AES and 32-/64-/128-bit blocks for RC5. Furthermore, we need a cryptographic primitive which is secure

in that it prevents an adversary to infer the original address by observing the translated address.

Our translation scheme is based on FF3 encryption, which is an instance of format-preserving encryption [36]. Equation 5.1 shows the performed translation, with k being a secret key known only to the ISP. This scheme generates a new source address and port tuple for every new flow. When the corresponding incoming packet arrives, the translation gateway performs the decryption, and then XORs the result with the source address and port of the incoming packet.

$$(saddr', sport') = FF3_k((saddr, sport) \oplus (daddr, dport)) \tag{5.1}$$

We utilize FF3 because it provides the required flexibility: it encrypts a plaintext of some format and length into a ciphertext of identical format and length, allowing us to shuffle variable-length address blocks. Furthermore, FF3 provides the security guarantees of conventional block ciphers [61] and is approved by NIST. Lastly, FF3 encryption is efficient; it is based on AES as the underlying block cipher, which is implemented in hardware even on commodity CPUs.

Processing Overhead. We quantify the processing overhead of our flexible address-translation scheme. We have implemented the stateless AHS service on the Data Plane Development Kit (DPDK) [13], running on a commodity server equipped with a 10 Gbps NIC and an Intel XEON E5 CPU. We evaluate three cases of shuffling a varying number of addresses: i) 2^{16} addresses in one /16 prefix, ii) 2^{24} addresses in one /8 prefix, and iii) 2^{26} addresses of the 4401 disjoint prefixes of an ISP (AS 4130). Furthermore, we evaluate forwarding performance for multiple packet sizes and for a representative mix of Internet traffic (IMIX [129] with 362 bytes avg. size); the baseline for our measurements is the performance of typical IP forwarding without additional processing.

Figure 5.3 shows the forwarding performance. For 64-byte packets, performance degrades by about 25% for shuffling an /8 and a /16 prefix. However, the decline is higher for the 2^{26} addresses because the blocks are disjoint and must be linearized. For larger packet sizes almost all cases perform optimally. The evaluation shows the efficiency of a single translation gateway—it can handle at line-rate a fully-saturated 10 Gbps link with typical Internet traffic patterns.

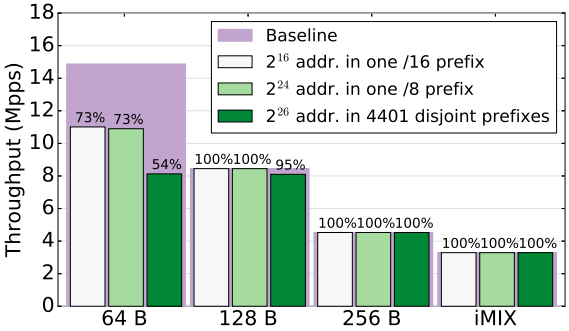


Figure 5.3: Forwarding performance of a translation gateway.

5.2.3.2 ISP-wide Secure Tunnels

We propose that ISPs set up pairwise encrypted tunnels using existing protocols, such as IPsec in tunnel mode [102] and the Resource Public-Key Infrastructure (RPKI) [46]. IPsec in tunnel mode creates a virtual private network between two remote networks: each network deploys an IPsec gateway and the exchanged traffic is tunneled as ciphertext between the gateways. The RPKI consists of publicly accessible repositories of resource certificates, which prove that an AS is the holder of a resource (e.g., a certain IP prefix).

Although we leverage existing technologies, there are multiple challenges with respect to scalability, performance, and security. First, since there are thousands of ASes, it is hard to manually configure pairwise tunnels at a large scale. Then, how can we automate tunnel establishment so that key negotiations are performed dynamically?

Second, given the volume of traffic that ISPs forward, it is a considerable overhead to encrypt/decrypt even a fraction of an ISP's traffic. Furthermore, using a single IPsec gateway to serve all traffic of all established tunnels with other ISPs is not possible. Therefore, we describe an intradomain architecture that is capable of supporting ISP-wide tunnels.

Dynamic Tunnel Configuration ISPs interested in deploying pairwise tunnels could initially coordinate and exchange required information manually. However, this approach does not scale as more and more ISPs start offering the service. To support automated tunnel configuration, two steps

are needed: i) discover deploying peers, and ii) establish a security association with the peer.

Peer Discovery. The first step is to enable ASes to advertise support for the ISP-wide tunnels to other ASes. To this end, we leverage resource certificates and RPKI to disseminate the additional information needed for ISP-wide tunnels.

A resource certificate verifies that an AS owns a certain prefix and is therefore authorized to make a BGP announcement for that prefix [87]. Typically, a resource certificate contains the AS number, the public key of the AS, and a list of prefixes that the AS owns; the certificate is signed by the private key of a regional or a local Internet registry. We augment such certificates with additional information for the tunnel establishment; such information includes cryptographic material for key exchanges, security parameters, and addresses of the IPsec gateways. Presence of this information indicates that the ISP supports the tunneling service.

The process of peer discovery is then performed as part of route-origin authorization [111]: when ASes receive BGP announcements from their peers, they consult RPKI and verify that the advertised prefix is legitimate through the corresponding resource certificate. At this stage an AS can further learn if the advertised AS supports tunnels.

Key Exchange. The second step for automated tunnel configuration is to establish a security association (SA). A SA is a structure that contains all necessary parameters for the corresponding tunnel: the tunnel endpoint addresses, the encryption and authentication algorithms, and the symmetric key used for encryption/decryption.

In order to set up a SA, IKEv2 [99] can be used. The protocol can perform mutual authentication between two IPsec gateways based on x509 certificates; it also negotiates the security parameters for a SA. To facilitate authentication and key exchange, we leverage again RPKI and the resource certificates: the published x509 resource certificate contains a public value that is used to perform an authenticated Diffie-Hellman key exchange and derive the symmetric key.

Intra-domain Architecture In addition to mechanisms for setting up tunnels, an ISP will need mechanisms inside its network to support encryption/decryption of tunneled traffic. We benchmark the operation of a sin-

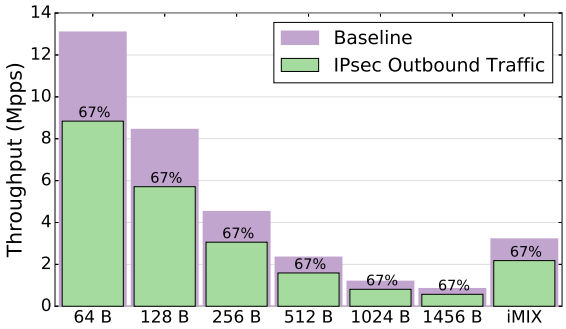


Figure 5.4: Forwarding performance of an IPsec gateway.

gle IPsec gateway, and we describe how to support multiple gateways with a low management overhead.

Gateway Evaluation. A deploying ISP may have to tunnel a considerable portion of traffic if many of its customers opt for the service. To put the processing overhead into context, we benchmark the performance of a gateway assuming it has a tunnel with every AS.

We use DPDK and run an IPsec gateway on the same commodity server. Then, we analyze a BGP routing information base[3] to generate realistic SAs. More precisely, we generate 56,223 SAs—one for every AS—and we construct over 200k routing policies so that each packet is forwarded over the correct tunnel based on the destination address. For encryption and authentication we use the AES block cipher in GCM mode with 128-bit keys [125].

Figure 5.4 shows the forwarding performance of the gateway for the outbound direction and for multiple packet sizes, including the IMIX (inbound direction has the identical performance); the performance baseline is forwarding without any additional processing. The results show that performance is at approximately 67% of the baseline for all cases. This constant performance decline is observed because as we increase the packet length the packet rate decreases for a fully utilized link. At the same time, the length of the plaintext to be encrypted increases so that these two factors cancel each other out. The experiment shows the worst case in

[3] http://data.ris.ripe.net/rrc06/2017.04/

that we have established the maximum number of tunnels. However, the result also indicates that ISPs may have to expand their infrastructure if many users opt for the service.

Supporting Multiple Gateways. An ISP will need multiple gateways to support the traffic demands. However, this raises a challenge if the gateways of the two ISPs have to maintain a mesh of tunnels between them.

We introduce the concept of a *logical* IPsec gateway which is decoupled from the underlying *physical* IPsec gateways. That is, an ISP advertises a single tunnel end point and only one IPsec tunnel is established per remote ISP. Then, the state of the tunnel is shared between all physical IPsec gateways so that they can perform identical operations.

We make the following design choices to realize a logical IPsec gateway. First, we define a *designated* IPsec gateway[4] that represents all physical IPsec gateways in an ISP and is identified by an IP address within the ISP's prefix blocks; the same address that is published in the resource certificate. The designated gateway establishes the ISP-wide tunnels with the designated gateways of other ISPs, and then it disseminates all necessary state to the physical IPsec gateways of the ISP. Second, we leverage IP anycast so that a single IP address is used for all physical IPsec gateways. This enables ISPs to support a tunnel end point from multiple locations and at the same time perform load-balancing among these locations by adjusting their intra-domain routing protocols.

5.2.3.3 ISP-level VPN

We propose that ISPs leverage their infrastructure to offer VPN services to customers of other ISPs.

As mentioned in Section 5.2.2.2, a VPN service consists of two main functionalities: 1) an encrypted tunnel that transfers packets between a VPN customer and a VPN gateway, and 2) address translation that replaces the customer's IP address with an IP owned by the VPN provider.

We have already addressed the challenges of the two functionalities, albeit in a different context. In Section 5.2.3.2, we have described how to set up ISP-wide tunnels using IPsec; the same approach can be used for the tunnel between a VPN customer and the gateway. There are various implementations of VPN tunnels (e.g., VPN over IPsec and VPN over

[4] Similar to designated routers in OSPF [130] and the IS-IS [89] protocols.

SSL), but they are conceptually the same. Furthermore, authentication, key exchange and tunnel configuration is typically performed through an application that is provided by the VPN provider and is installed on the customer's device.

In Section 5.2.3.1, we have shown how to perform ISP-level address translation. A VPN provider can use the same approach to multiplex multiple customers behind the source-port number and the host portion of its IP address block.

5.2.4 Composition of Privacy Services

In Section 5.2.2, we described how three building blocks can offer meaningful privacy services. Here we explain how composition of the proposed services can create additional privacy services.

AHS+STS Composition. Consider a host that subscribes to the address-hiding service and the secure tunneling service of its ISP. This combination provides additional privacy benefits that are not offered by these services in isolation by enabling: i) to hide also the host's address from the destination ISP and host (not offered by secure tunneling), and ii) to hide the transport-layer header and payload from all transit ISPs (not offered by address hiding).

Privacy Domains. Three privacy domains are formed. Domain *A* consists of the source ISP, and the host shares all its information. Domain *B* consists of the destination host and ISP, and the host shares all its information except for the source address, which is instead revealed at the AS granularity. Domain *C* consists of all transit ISPs, and the host shares the source and destination addresses at the AS granularity.

IVS+STS Composition. Another meaningful service composition is the following: a host buys the VPN service from a remote ISP and also the secure tunneling service from that ISP. Again this combination offers privacy benefits that cannot be achieved by a single service: i) the transport-layer header and payload are hidden from the host's ISP (not offered by secure tunneling), and ii) the host hides its payload from all transit ISPs between the VPN provider and the destination ISP and host (not offered by the VPN service).

Privacy Domains. Specifically, it creates the following four privacy domains. Domain *A* consists of the source ISP, and the transit ISPs to the

remote VPN ISP, and the host shares only his address with this domain. Domain *B* consists of the remote VPN ISP, and the host shares all information. Domain *C* consists of the transit ISPs between the remote VPN ISP and the destination ISP; with this domain, the host shares only the destination address at the AS granularity. Finally, Domain *D* consists of the destination host and ISP; with this domain, the host shares the payload, the transport header, and the destination address.

A sequence of IVS Composition. A Tor-esque privacy service can be created by using a sequence of IVSes, where each IVS acts as a Tor relay. Specifically, a host creates an onion by encrypting his message in layers and forwards through a series of IVSes, each of which peels off an encryption layer and forwards to the next IVS.

Privacy Domains. This privacy service creates the same privacy domains that are described for Tor in Section 5.2.1.

Challenges. We envision that an IVS is a paid service that requires explicit subscription; hence, a client needs to authenticate to all IVSes that it uses. This authentication leads to a privacy implication: all IVS-offering ISPs that are part of the Tor-esque privacy service can identify the sender based on client's authentication credentials.

One possible approach to solve this problem is based on anonymous credentials. In this approach, ISPs could form an alliance and allow a host who subscribes to any member ISP to use the privacy services offered by all member ISPs. The alliance provides hosts with anonymous credentials that prove that the host is authorized to use the privacy services without revealing any further information about the host. Then, hosts use their anonymous credentials to authenticate to the IVSes.

5.2.5 Deployment Incentives for ISPs

We believe that our proposed services are in line with market incentives of today's Internet. A major deployment hurdle for many proposals is the lack of incremental deployability so that the proposals are valuable if all, or almost all, ISPs adopt; there are no benefits for first-movers, leading to a chicken and egg problem. This is not the case for two of our three proposed services: the address-hiding service and the VPN service can be offered to end users without requiring global adoption of new protocols. The secure-tunneling service has a higher deployment barrier in that it

requires coordination. However, setting up a tunnel requires coordination only between two ISPs—not universal coordination—and we believe there are strong incentives for adoption: on one hand, large content providers are concerned about large-scale surveillance that degrades their customers' privacy. On the other hand, residential ISPs can offer value-added services to their customers by setting up such tunnels with large content providers.

Chapter 6
Related Work

In this chapter, we review the related work in the following relevant areas:

- Section 6.1: Balancing Source Accountability and Privacy
- Section 6.2: Source Accountability
- Section 6.3: Communication Privacy
- Section 6.4: Incremental Deployment for FIAs

6.1 Balancing Source Accountability and Privacy

Accountable and Private Internet Protocol (APIP) [134] 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*, which is the trusted third-party entity 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 has a few limitations. First, APIP's notion of privacy is limited to sender-flow unlinkability. It does not provide other unlinkability properties, such as flow-packet unlinkability, since flow identifiers are present in the packets. Moreover, APIP leaves data privacy and the associated challenges (e.g., key distribution, management, and establishment) unaddressed. Second, 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 information of those packets has been "whitelisted". Specifically, 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 [134]). Third, 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.

Persona [122] is another proposal that simultaneously provides privacy and accountability at the network layer. To guarantee the privacy of the hosts, each router on the forwarding path to the destination translates the source address in the packet to one of its own. The translation provides two privacy benefits. The source addresses in packets do not identify the sending host; and, as a packet traverses further away from the sender, the anonymity set for the sender increases. For source accountability, Persona reveals the identity of a sender when the sender misbehaves. Specifically, the procedure for identifying the misbehaving sender is defined as follows. Starting from the router that is immediately connected to the destination, the routers sequentially reveal their translation of the source addresses.

Persona has a few limitations. First, it breaks communication on the Internet since it breaks the notion of flow and prevents the destination from demultiplexing connections. Moreover, hosts in Persona may suffer from severe packet re-ordering since each packet could take a different router-level path to the destination. Second, Persona assumes that source address spoofing is not possible without specifying how the architecture attributes packets to their senders.

6.2 Source Accountability

6.2.1 Source Authentication

Ingress Filtering. Ingress Filtering provides a weak form of source authentication by dropping packets with spoofed source addresses. On a high-level, ingress filtering asks the following question to determine if an address has been spoofed: "Can a packet with its source address arrive at this interface of this router?" If the answer is no, then the source address is considered to be spoofed and the packet is dropped.

The question is answered by creating an access list, and there are various ways to create the access list. In one approach, a network operator manually creates an ingress access list that contains all acceptable IP prefixes. Alternatively, the access list can be created dynamically. uRPF [31] generates the access list based on Forwarding Information Base (FIB) at a router; SAVE [112] based on source addresses in packets that a router forwards; HCF [93] based on TTL values in forwarded packets; DFP [140]

based on AS-level routes from source ASes to destination ASes; IDFP [57] based on BGP update messages.

Ingress filtering protects source addresses only at a domain granularity and cannot prevent spoofing by an adversary that resides in the same domain as its victim.

Interactive Challenge-Response Authentication. A router could ask a more direct question than the one raised for ingress filtering:"Did you (the source host) send this packet?" One approach that is based on this question is an interactive challenge-response authentication protocol. In this protocol, a router that receives a packet presents a question to which the source host must correctly answer. Specifically, for each received packet, a router generates a cryptographic nonce (e.g., a MAC over a nonce and the hash of the packet using its secret key) as a challenge. Then, the source generates a response that uses the challenge as the input and is attributable to the source by the router. For example, a response could be a MAC over the cryptographic nonce using a symmetric key that the source shares with the router; alternatively, it could be a digital signature over the challenge using the source's private key. Regardless of the implementation, we call this approach *interactive* since a challenge is dynamically created only when a router receives a packet from a host.

AIP [25] uses an interactive challenge-response protocol with responses based on digital signatures. Specifically, a router constructs a cryptographic nonce by computing a MAC over a source's packet using a symmetric key that is only known to the router. Then, the source host responds by signing the nonce using the private key of which the corresponding public key (more precisely its hash) is the source address in the packet. APIP [134] uses a similar construct but with two differences: 1) the challenge is raised to the accountability delegate of the source host, instead to the host himself; 2) the delegate—not the source host—generates a response with its private key.

An interactive Challenge-response protocol has one disadvantage; it requires additional latency due to the challenge-response protocol. To overcome this disadvantage, both AIP and APIP additionally use whitelist-based caches to amortize the latency penalty. That is, once a flow is authenticated, packets of the flow are no longer verified for some time. However, this approach sacrifices security over performance, since it precludes ev-

ery packet from being accounted for in the network and creates an attack window.

Non-interactive Challenge-Response Authentication. Making the challenge-response protocol non-interactive eliminates the additional latency due to the communication overhead for the challenge-response protocol. To make the protocol non-interactive, the challenge is predetermined, and a source host includes the response into his packet.

Many works, including APNA (Chapter 2), use this approach, and they all use a similar challenge-response construct. A predetermined challenge is based on the invariant fields of the packets (e.g., source, destination addresses, payload), and a response is a MAC over the challenge using a symmetric key that is shared between a source and a verifier. In APIP [134], the construct is used between source host and his accountability delegate so that the delegate can vouch for host's packets. In Packet Passport [115], the construct is used between the source, transit and destination ASes. The source AS constructs a sequence of response (i.e., MACs) one for each AS, and the transit and the destination ASes verify the corresponding responses. OPT [103] generalizes this model and allows each router on the communication path to authenticate hosts' packets. To this end, OPT proposes Dynamically Recreatable Keys (DRKeys) that obviate per-host or per-session state at each router that arises due to the shared keys needed for source authentication.

AaaS [38] also uses a similar challenge-response construct but defines three types of verifiers that perform source authentication for different purposes. The source AS authenticates a packet to the source host to prevent source address spoofing within the AS; the transit and the destination ASes authenticate the packet to the source AS to prevent address spoofing across ASes; and, the accountability agent of the source host authenticates the packet to the host to provide accountability services.

AaaS defines a new entity—that is not the ISP of the source—to provide accountability service. The authors separate the role of an accountability agent from ISPs based on an assumption that ISPs are reluctant to act as accountability for their customers due to conflicts of interest. For example, ISPs are not willing to take actions against its paying customers even if they misbehave. However, we do not agree with this assumption. Researchers have argued that ISPs should assume an active role in keeping the Inter-

net clean [113, 137, 156]; and, recently, ISPs have started assuming that role [53, 118, 121]. Furthermore, ISPs have been disconnected from the Internet due to their misbehaving hosts [68, 104, 105].

6.2.2 Blocking Unwanted Traffic

In Chapter 2, we describe the shutoff functionality that allows the receivers to suppress unwanted communication at the senders' ISPs. In this section, we review related works that take two different approaches to allow receivers to block unwanted traffic: a whilelist approach via network capabilities and a blacklist approach via network filters.

Using Network Capabilities. A Network capability is an authorization token that approves an entity to perform a certain action. For example, a server provides network capabilities to the clients to allow the clients to send packets to the server. This application of network capabilities has been proposed to limit DoS attacks against the receivers [26, 174, 176]. In these works, the network constructs a capability that the receiver provides to the sender. Then, the sender inserts the provided capability into his packet to the receiver; and, the network forwards the packet only if the packet contains a valid network capability.

Using network capabilities efficiently reduces unwanted traffic (e.g., DoS traffic), since the senders must get the consents of their receivers before sending any packet. However, this approach comes with two disadvantages. First, it introduces the initial latency for acquiring the receivers' consents. Second, it introduces a new attack window: denial of capability attacks [28]. That is, the adversary floods the destinations with capability requests.

Using Network Filters. Another approach is based on a blacklist. That is, senders are allowed to send packets to the receivers without requiring the receivers' consents. Then, a receiver that receives unwanted traffic from a sender installs a filter near the sender to suppress such traffic. More precisely, a destination sends a filter request (i.e., a shutoff request) for an unwanted flow, and the source validates the request (e.g., if the destination is authorized to make such a request) and blacklists that flow to suppress packets of that flow from leaving the source.

There are two locations at the source at which filters can be installed. In host-based approaches [25, 161], the filters are installed at the send-

ing hosts, e.g., using smart NICs. While this approach does not allow unwanted traffic to even enter the source network, this approach is problematic for compromised hosts as they could disable the filtering functionality. On the other hand, the network-based approaches [28, 116, 134], which install filters at a network infrastructure (e.g., at access or border routers), do not have such a problem. However, since routers need to maintain the filters, routers are susceptible to state exhaustion attacks that install a large number of filters.

In APNA, we use the network-based filtering approach and mitigates the state exhaustion attack by the following two design decisions. First, an EphID—not a specific flow—is blocked when there is a shutoff request. Blocking an EphID can create a collateral damage to a source host if he uses the EphID for multiple communications (e.g., using as a per-application EphID). The fear of collateral damage provides an incentive for the hosts to be *good citizens* to minimize filtering requests against them. Second, since EphIDs are bound to HIDs, the network can revoke the HID of an offending host—instead of blocking his individual EphIDs—if there are too many shutoff requests against the host.

6.3 Communication Privacy

Communication on the Internet leaks information about the communicating users, and these information can be used to build a history of users' communication activities. For example, the addresses in the network headers allow one to identify with whom users are communicating. Moreover, a similar observation can be made from the content of the communication by observing the packet payloads. In this section, we review related works on protecting the network header and the packet payloads.

6.3.1 Protecting the Content of Communication

Farrell and Tschofenig [66] 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 [101] proposes pervasive encryption as a countermeasure against pervasive monitoring. In a related effort, the Let's Encrypt[1] organization encourages the use of encrypted web traffic by issuing free TLS

[1] https://letsencrypt.org

certificates for web servers. Moreover, MinimaLT [144] proposes an architecture that supports pervasive data encryption and achieves PFS without sacrificing connection latency for communications.

On a related note, researchers have proposed opportunistic encryption (OE) [42, 43, 59, 65, 135, 153] to make it difficult for adversaries, such as governments, to perform pervasive monitoring. OE does not prevent active adversaries that perform man-in-the-middle attack from eavesdropping on communications, since OE does not require authentication between communicating entities. However, OE does make pervasive monitoring more difficult and expensive, since an adversary must perform a man-in-the-middle attack for each communication that he wants to eavesdrop.

In line with the trend for data privacy, our proposal proposes (authenticated) encryption as a fundamental design tenet of the network layer and additionally proposes a concrete solution for key distribution, establishment, and management. However, our position on network-layer encryption is different from the proponents of pervasive encryption. In our view, the architecture should allow users to decide if the network-layer encryption is necessary for their communication instead of mandating encryption for all communication.

6.3.2 Protecting Privacy Leak from the Network Headers

In this section, we review three approaches to prevent or limit an adversary from establishing a history of users' activities from the networks headers in the packets.

6.3.2.1 Removing Source Addresses.

Removing source addresses from the network header prevents an adversary from identifying the senders. However, this approach creates a challenge for bidirectional communication as the receiver does not know the address to which to send a reply.

Researchers have proposed various methods to allow bi-directional communication without source addresses in the packets. In NDN [90], packets leave a trail of "breadcrumbs" on the routers that they traverse, and the reply packets follow back the breadcrumbs to the senders. In LAP [85] and Dovetail [159], the network header contains a path from the source to the destination instead of their respective addresses. Then, the destination reverses the path to send a packet to the source. In APIP [134], the address

of the sender is included in the packet payload so that only the destination can identify the address.

6.3.2.2 Using Multiple Source Addresses

Alternatively, different source addresses could be used for different communications to limit the amount of information that can be linked by a single address. AHP [149] proposes an ISP-wide NAT that chooses a different source address for each outgoing flow. RFC 4941 [133] specifies a privacy extension for IPv6 that allows a host to generate multiple IPv6 addresses by choosing multiple unique values for the interface identifier (i.e., the lower eight bytes of the IPv6 address). Han et al. [79] proposes a pseudonym-based architecture where ISPs provide multiple pseudonyms to their hosts; a pseudonym is an IPv6 address with a cryptographically generated interface identifier.

Sakurai et al. [157] proposes communication based on one-time receiver addresses, which are IPv6 addresses with cryptographically generated interface identifiers. The interface identifiers are generated using a hash-chain with a symmetric key that is shared between a sender and a receiver; this construction allows the sender and the receiver to coordinate the one-time addresses. Although this work is closely related to OTA (Chapter 3), they fall short of achieving flow-packet unlinkability since only the destination address is changed for each transmitted packet.

6.3.2.3 Using Detours

Taking a detour through relays prevent an adversary from identifying the sender and the receiver at the same time (i.e., it provides sender-destination unlinkability). Anonymity systems are based on the idea of detours, and we broadly classify anonymity systems based on the amount of routing inflation that the systems create.

low-stretch anonymity systems. The simplest systems take one detour through a single anonymizing relay. Virtual Private Network (VPN) services (e.g., expressvpn.com) and anonymous proxies (e.g., anonymizer.com) are examples of such systems. In these systems, a host encrypts and sends a packet to the relay. Then, the relay decrypts the packet and sends the packet to the host's intended destination using the address of the relay as the source address. These systems are lightweight and incur low-latency

since only a single detour is made; however, they require users to trust the relay since it can identify the sender and destination at the same time.

high-stretch anonymity systems.

Onion Routing. If a user cannot trust a single relay, he can forward his packets through a series of relays. That is, the user encrypts his message in layers and forwards through a series of relays, each of which decrypts, strips off a layer and forwards to the next relay. This technique of using a series of relays is called onion routing [151]. In onion routing, a relay only knows the previous and the next relay; hence, none of the relays can identify the sender and destination at the same time.

The most popular instantiation of onion routing is Tor [56], which aims at providing low-latency communication. However, its design to provide low-latency communication also makes it vulnerable against traffic analysis attacks [162, 168, 169, 171, 179].

Furthermore, Tor inevitably suffers from high path inflation since it is implemented as an application-layer overlay. That is, an end-to-end path through the Tor network can be much longer than the end-to-end path through the Internet. Some proposals have addressed this limitation by deploying the anonymity systems in the network. That is, routers in ASes act as relays, and sending packets through the relays is equivalent to sending them through the Internet without the privacy protection. Examples of such anonymity systems include HORNET [49], LAP [85], Dovetail [159] and Tor instead of IP [114].

Mix Networks (MixNets). To increase resilience against traffic analysis, a relay can additionally batch and permute the order of packets to hide the correspondence between inputs and outputs to a relay. Such a relay is called a mix, and it has been first described by Chaum in his seminal paper [48]. Since then, many mixed networks have been proposed for different applications. For example, mixmaster [128], babel [77], and mixminion [54] have been proposed for the anonymous remailer services, and ISDN-mixes [146], Real-time Mixes [91] and Web Mixes [41] for telecommunication networks and web-browsing on today's Internet. Recently, Aqua [108] and Herd [107] have been proposed for anonymous peer-to-peer and VoIP applications, respectively. The two works exploit the distinct traffic patterns for their respective applications, to design a high-bandwidth (Aqua

for p2p applications) and low-latency (Herd for VoIP applications) mix networks.

6.4 Incremental Deployment for FIAs

Over the past 15 years, there have been numerous FIA proposals that take on various limitations of today's Internet; and most of them also describe how their architectures can be deployed in today's Internet. Although the exact details on how the architectures can be deployed differ from one work to another, they are all based on the same approach—deploy as an overlay over today's Internet.

In this section, we highlight deployment studies for two specific FIAs—NDN [90] and XIA [78]—instead of creating a long-list for all FIA proposals that have discussed deployment. We focus on those two architectures for two reasons. They are two of the prominent FIA projects that were sponsored by the NSF FIA project;[2] more importantly, the two FIAs show how the design of an architecture affects its deployability.

NDN Deployment. Deploying NDN on today's Internet is challenging. First, NDN's routing and forwarding are incompatible with today's IPv4 network. Second, the benefits of NDN (e.g., efficient content delivery with low latency) are problematic in early stages of deployment. This is because the benefits of NDN are known to materialize only when the network can cache contents pervasively, but pervasive caching is difficult in early stages of deployment since only a few NDN routers would be deployed.

Researchers have adapted NDN to be more amenable for deployment, and one example is idICN [67]. The authors of idICN start by analyzing the benefit of pervasive caching. To this end, the authors compare the benefit of pervasive caching and caching only at the edge networks (e.g., stub ASes) and concludes that edge-only caching is nearly as good as pervasive caching. Then, based on their conclusion, the authors describe idICN that implements NDN functionalities (e.g., content caching, name resolution, content verification) only at the edges of the network (e.g., stub ASes and end-hosts). This approach is easier to deploy than the original architecture because the network edges are easier to change than the core, and the deploying entities benefit from deployment. Specifically, adopting end-

[2] www.nets-fia.net

users retrieve contents with lower latency due to content caching, and deploying ISPs generate revenue from end-users who use idICN.

Another work is hICN [4] that focuses on adapting NDN to be compatible with IPv6 networks. To this end, hICN maps the names of the contents to IPv6 addresses, and defines NDN packets (e.g., interest and data packets) as IPv6 packets. This design allows IPv6 routers and NDN routers to coexist so that NDN routers can be gradually introduced into an IPv6 network. In a hICN network, the IPv6 routers in the network treat all packets (including the NDN packets) as IPv6 packets and simply forwards them based on their IPv6 headers while the NDN routers recognize NDN packets and process them accordingly.

XIA Deployment. Unlike ICN, XIA can be incrementally deployed without modifying the architecture because its design allows various types of communication (e.g., host-based, content-based communications) to coexist in the network. To this end, XIA introduces two architectural elements. First, XIA defines principal types that users use to specify their communication intent (e.g., content retrieval), and that specify forwarding semantics for the network. Second, XIA addresses support fallback addressing. Fallback addressing is routing information that a router can use if it cannot understand the destination information in the packet, e.g., because the destination information includes a new principal type.

For deployment over today's Internet, XIA defines a new principal type, 4ID that encodes IPv4 addresses [75]. When a host in one XIA island wants to communicate with another host in another island that is separated by today's Internet, the source host specifies the 4ID of the ingress XIA router as a fallback address. Then, the egress XIA router in the source island tunnels host's packet to the destination island by using the IPv4 address that is encoded into the fallback address.

Deployment for Generic Architecture. Mukerjee et al. [131] does not study incremental deployability of a specific architecture, but the authors study the design issues that arise when incrementally deploying an architecture: bridging two islands of a new network across an old network. In such a case, communication between two hosts in two different islands must go through an egress point at the border between the source island and the old network, and an ingress point at the border between the old network to the destination island. The authors analyze the design space in

selecting these two points with respect to performance (e.g., latency) and flexibility (e.g., recovery from a failure of ingress or egress points).

Chapter 7
Conclusion and Future Work

Both source accountability and privacy are important properties for the Internet. However, the conflicting nature of the two properties makes it difficult to simultaneously support both properties. To date, the research community has predominantly investigated approaches that favor either privacy or accountability, offering one at the expense of the other. To our knowledge, APIP [134], is the only major proposal that has aimed to find a balance between the two properties at the network layer; however, APIP presents a high-level design and leaves details unspecified.

This dissertation investigates the design and deployment of the Accountable and Private Network Architecture (APNA) that provides source accountability guarantees and is privacy-preserving.

Chapter 2 specifies the base architecture of APNA, which can be summarized by the following two design decisions. First, we enlist ISPs as the trusted third-party entity to balance accountability and privacy. As an accountability agent, an ISP establishes a strong notion of host identity of its host and a strong link between the identity of the source and the sent packet. As a privacy-broker, an ISP issues privacy-preserving addresses (i.e., EphIDs) that hosts use as source addresses. Furthermore, an ISP acts as a certificate issuer, issuing certificates that certify the binding between public keys of its hosts and EphIDs. The certificates are used to securely establish shared symmetric keys that are used to hide communication between hosts, even against their ISPs.

Second, using EphIDs as network addresses allows us to design an efficient data-plane that provides both source accountability guarantees and privacy-preserving communication. This is due to how EphIDs are constructed and used; ISPs create EphIDs by encrypting the identities of their hosts using secret keys that are only known to them, and ISPs issue multiple EphIDs to their hosts. This EphID construction allows only the issuing ISPs to identify the identities of the hosts from EphIDs, and the ISPs make the association without maintaining per-EphID state; the stateless binding allows us to design efficient source authentication and routing mechanisms that do not require per-EphID state. Furthermore, since a host can obtain

multiple EphIDs, the host can control the amount of traffic that can be linked to a common sender by using different EphIDs for different communication, hence enhancing communication privacy.

Chapter 3 pushes the idea of using EphIDs at different granularities to the extreme—use a single EphID to either receive or send a single packet. That is, EphIDs are used as per-packet One-Time Addresses (OTAs). Communication based on OTAs (if designed correctly) achieves an interesting property—*flow-packet unlinkability*: by observing packets of any number of flows, the packets are no more and no less related to any flow after the observation than they were before the observation. Communication that achieves flow-packet unlinkability dispels many sophisticated attacks [162, 168, 169, 171, 179] that use flow information within packets. We extend APNA to support communication that satisfies flow-packet unlinkability.

Chapter 4 addresses APNA's vulnerability against replay attacks. To emphasize the severity of replay attacks, we identify a new attack—the router-reflection attack—that attacks a remote geographic region of the Internet by simply replaying packets from compromised routers. The router-reflection attack shows that a replay suppression functionality is necessary within the network to protect itself, and the functionality should not be entirely delegated to end-hosts. In addition, it challenges whether the end-to-end argument applies to duplicate packet suppression. Then, we describe a lightweight in-network replay suppression mechanism.

Chapter 5 investigates two deployment strategies for APNA. Specifically, we describe how an overlay network of APNA could be created over today's Internet. In addition, we propose a new integrated approach to FIA deployment, which can be summarized by the following phrase: *think revolutionary, act evolutionary*. This approach considers the FIA design process as a platform to identify new ideas, and integrates such ideas on today's Internet using well-established technologies. We apply this deployment approach to APNA by realizing the notion of user-defined privacy on today's Internet. To this end, we propose a new concept, *privacy domain*, which allows users to express their privacy requirements; then, we identify ISP-level privacy services that can be used to construct users' privacy domains.

7.1 Future Work

The following are future works that we have identified.

Formal Verification for APNA. APNA relies on cryptographic primitives to provide source accountability guarantees and support privacy-preserving communication. To avoid introducing security vulnerabilities, we only use cryptographic primitives that are secure and design each protocol while anticipating possible attacks against the protocols. Furthermore, we specify an adversary model and qualitatively analyze the security of our architecture against the adversary model. However, we still cannot conclusively say that our architecture is free from security vulnerabilities; formal protocol verification would provide that answer.

Support for Privacy-preserving Transport Protocols. In our work, we do not consider side-channel attacks (e.g., timing anlysis) that can violate privacy guarantees. We have not considered side-channel attacks since we believe that the architecture should provide only the basic building blocks to achieve host privacy at the network layer and stronger privacy guarantees (e.g., resiliency against timing analysis) should be provided by protocols at a higher layer (e.g., transport protocol).

It would be interesting future work to design privacy-preserving transport protocols that can combat side-channel attacks, and we identify two interesting issues when combining such transport protocols with today's transport protocols (e.g., UDP, TCP, SCTP [166]). First, we need to consider how privacy-preserving transport protocols can be integrated with today's transport protocols. For example, should the privacy-preserving transport protocols be designed modularly and then allow users to combine them with today's transport protocols? Alternatively, should the privacy-preserving transport protocols directly implement the functionalities of today's transport protocols?

In addition, we need to study the performance implication when a privacy-preserving transport protocol is combined with today's transport protocol (especially for TCP or SCTP). This is necessary since a protocol that alters packet transmission times (e.g., one that uniformizes inter-packet times) may have a significant impact on TCP performance.

Support for Privacy-preserving Low-latency Inter-domain Mobility/Multi-homing. APNA, as described in the dissertation, cannot support mobility[1] across ISPs because EphIDs are valid only within an ISP. When a host moves from one domain to another, the host receives new EphIDs from his new domain, and all EphIDs from the previous domain become invalid. To support mobility, the host needs to inform his peer about new EphIDs; but, APNA does not specify such a mechanism.

One approach to support mobility is to follow the approach of QUIC [148]. In QUIC, a client and a server specify a connection UUID that remains constant during the lifetime of a connection. When the client moves from one domain to another, the client informs its new IP address to the server by using the connection UUID. A similar method can be used in APNA to support mobile hosts. When a mobile host moves from one ISP to another, he can update the peer about the new EphID using the connection UUID. However, this approach has a privacy implication. Since a connection UUID must be visible in the packets, an observer can track the movement of a host using the connection UUID.

One could offer the following as the solution to this privacy implication. The two hosts first establish a new secure connection (using the procedure in Section 2.3.4.1), with the mobile one using a new EphID. Then, the mobile host sends a message to the other host to indicate that the new connection is a continuation of the previous connection. This approach eliminates the privacy implication since the connection UUID would be encrypted using the shared symmetric key for the new secure connection. However, this approach incurs two additional RTTs: one RTT is needed to establish a new connection, and another RTT is needed to associate the new connection to the previous connection. Designing a privacy-preserving low-latency inter-domain mobility mechanism is not a trivial problem.

Privacy-as-a-Service. Another future work is to develop a platform (shown in Figure 7.1) to realize the notion of *user-defined privacy* on today's Internet. The goal of the platform is to create *privacy circuits* based on users' communication requirements, which would be expressed in terms of privacy and performance requirements.

[1] We discuss only in the context of mobility; the argument also applies to multi-homing since conceptually they are equivalent.

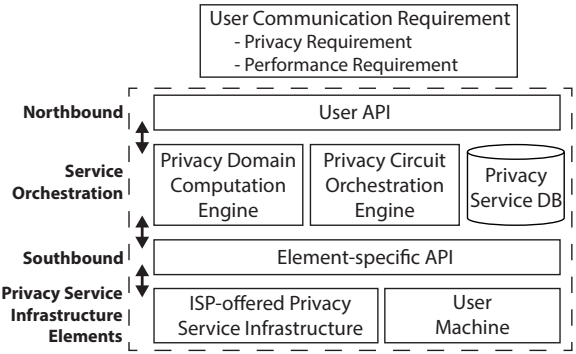


Figure 7.1: A platform for Privacy-as-a-Service.

The platform uses the privacy services (See Section 5.2.2.2) offered by the network to create the privacy circuits based on users' communication requirements. To this end, the two engines in the platform (i.e., the privacy domain computation engine and the privacy circuit orchestration engine) jointly compute the privacy circuits. Specifically, the privacy domain computation engine translates the user's high-level privacy requirements to more actionable technical requirements by constructing privacy domains that describe what privacy-sensitive information (e.g., source address) are revealed to which entities (e.g., ISPs). The privacy circuit orchestration engine creates a set of actionable instructions to create a privacy circuit that realizes the computed privacy domains. Then these instructions are delivered to the ISPs' privacy service infrastructures and the host's computer to configure the computed privacy circuits.

We identify the following specific future works towards developing the Privacy-as-a-Service platform:

- **Privacy Services.** In this dissertation, we have identified four privacy primitives: multiple IPv6 addresses through the IPv6 privacy extension [133], and the three ISP-offered services described in Chapter 5, i.e., ISP-level Address Hiding Service, ISP-wide Secure Tunneling Service, and ISP-level VPN Service. It is an interesting future work to identify other privacy primitives.

- **Algorithms for the two engines.** Efficient algorithms must be developed for the privacy domain computation engine and the privacy circuit orchestration engine.
- **Privacy Abstraction.** A study is needed to find an appropriate abstraction to express users' privacy requirements. For example, we need to investigate if a user would be capable of or interested in making privacy statements such as "I want to avoid exposing my IP address against AS X". Alternatively, it may be appropriate to provide a simple API that provides a few pre-configured options (e.g., privacy-critical, performance-critical and balanced).
- **User Communication Requirements.** We believe that users would be interested in expressing their communication requirements from performance (e.g., latency, bandwidth) and privacy perspective; however, a further study is needed to determine the criteria that users would use to describe their communication requirements.
- **Privacy Circuit Management.** A user can create multiple privacy circuits since users would have different requirements for different communications. Hence, the platform must allow a user to choose an appropriate circuit for his particular communication. A study is needed to determine how this interaction should be made. One possible approach is using a similar approach to the permission control in mobile devices. When an application is installed or executed, the platform asks the user to select a privacy circuit for the application.
- **Business Model.** We expect users to subscribe and pay to use the privacy services of the ISPs. If a subscription is made on a per-ISP basis, a user needs to individually subscribe to multiple ISPs when creating a privacy circuit that uses privacy services of many ISPs (e.g., a Tor-esque privacy circuit using multiple ISP-level VPN services). In such a case, a per-ISP subscription may not scale if users need to use privacy services of many ISPs. Hence, a different business model may be necessary. For example, ISPs could form an alliance and allow a user who subscribes to any of the ISPs in the alliance to use the privacy services offered by all member ISPs.

Bibliography

[1] CAIDA: Looking Glass API. http://goo.gl/AKQcd9.

[2] Configuring Per-Packet Load Balancing. http://goo.gl/ZO4LiS.

[3] How Does Load Balancing Work? http://goo.gl/SVbYM9.

[4] Mobile Video Delivery with Hybrid ICN: IP-integrated ICN solution for 5G. https://goo.gl/KLd8YT.

[5] RIPE Atlas. http://atlas.ripe.net.

[6] SDN Security Challenges in SDN Environments. http://bit.ly/2av8huG.

[7] The CAIDA UCSD Anonymized Internet Traces 2015-050615. http://goo.gl/WmItAH.

[8] CRIME Attack Uses Compression Ratio of TLS Requests as Side Channel to Hijack Secure Sessions. http://bit.ly/2rn7QbT, 2012.

[9] A Perfect CRIME? Only TIME Will Tell. http://ubm.io/2rn5qtU, 2013.

[10] The BREACH Attack. http://www.breachattack.com/, 2013.

[11] The Heartbleed Bug. heartbleed.com, 2014.

[12] Cisco Routers Compromised by Malicious Code Injection. http://goo.gl/oWBtF6, Sep 2015.

[13] Data Plane Development Kit. http://dpdk.org, Sep 2015. Retrieved on 1/2016.

[14] eBACS: ENCRYPT Benchmarking of Cryptographic Systems. http://bench.cr.yp.to/supercop.html, 2015. Retreived on 4/2016.

[15] Juniper ScreenOS Authentication Backdoor. https://goo.gl/umV2gD, Dec 2015.

[16] Snowden: The NSA planted backdoors in Cisco products. http://goo.gl/xwdFW2, May 2015.

[17] Spirent SPT-N4U-220 Chassis. http://goo.gl/X4gbeI, Sep 2015. Retrieved on 4/2016.

[18] The Copyright Alert System. http://goo.gl/UsilIf, Jan 2015. Retrieved on 12/2015.

[19] BackConnect's Suspicious BGP Hijacks. http://bit.ly/2rnhBa8, 2016.

[20] Large Hijack Affects Reachability of High Traffic Destinations. http://bit.ly/2qMW6Rm, 2016.

[21] Iran's Porn Censorship Broke Browsers as Far Away as Hong Kong. http://bit.ly/2s4E90s, 2017.

[22] Russian-controlled Telecom Hijacks Financial Services' Internet Traffic. http://bit.ly/2pp44fp, 2017.

[23] SSL by Default Usage Statistics. http://bit.ly/2rnj1S7, 2017.

[24] SSL Pulse. https://www.trustworthyinternet.org/ssl-pulse/, 2017.

[25] D. G. Andersen, H. Balakrishnan, N. Feamster, T. Koponen, D. Moon, and S. Shenker. Accountable Internet Protocol (AIP). In *Proceedings of the ACM Conference on SIGCOMM*, 2008.

[26] T. Anderson, T. Roscoe, and D. Wetherall. Preventing Internet Denial-of-Service with Capabilities. In *Proceedings of the ACM Workshop on Hot Topics in Networks (HotNets)*, 2003.

[27] Andre. What Everybody Ought to Know About HideMyAss. https://goo.gl/GYKLjH, Feb 2016.

[28] K. Argyraki and D. R. Cheriton. Active Internet Traffic Filtering: Real-Time Response to Denial-of-Service Attacks. In *Proceedings of the USENIX Annual Technical Conference (ATC)*, 2005.

[29] ARIN. Resource Public Key Infrastructure. http://bit.ly/1EJCQoT, Jan 2015. Retrieved on 2/2016.

[30] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira. Avoiding Traceroute Anomalies with Paris Traceroute. In *Proceedings of the ACM Conference on Internet Measurement (IMC)*, 2006.

[31] F. Baker and P. Savola. Ingress Filtering for Multihomed Networks. RFC 3704 (Best Current Practice), Mar. 2004.

[32] J. Bamford. The NSA Is Building the Country's Biggest Spy Center (Watch What You Say). https://goo.gl/2XC2Sc, Mar 2012.

[33] C. Basescu, R. M. Reischuk, P. Szalachowski, A. Perrig, Y. Zhang, H.-C. Hsiao, K. A., and U. J. SIBRA:Scalable Internet Bandwidth Reservation Architecture. In *Proceedings of the Symposium on Network and Distributed System Security (NDSS)*, 2016.

[34] M. Bellare, J. Kilian, and P. Rogaway. The Security of the Cipher Block Chaining Message Authentication Code. *Journal of Computer and System Science*, 61(3), 2001.

[35] M. Bellare and C. Namprempre. Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. In *Advances in Cryptology—ASIACRYPT*, volume 1976. Springer, 2000.

[36] M. Bellare, T. Ristenpart, P. Rogaway, and T. Stegers. Format-Preserving Encryption. In *Proceedings of the Workshop on Selected Areas in Cryptography*, 2009.

[37] S. M. Bellovin, D. D. Clark, A. Perrig, and D. Song. A Clean-Slate Design for the Next-Generation Secure Internet. Technical Report GDD-05-05, NSF Workshop Report, 2005.

[38] A. Bender, N. Spring, D. Levin, and B. Bhattacharjee. Accountability as a Service. In *Proceedings of the USENIX Workshop on Steps to Reducing Unwanted Traffic on the Internet (SRUTI)*, 2007.

[39] D. J. Bernstein. Curve25519: new Diffie-Hellman speed records. In *Public Key Cryptography (PKC)*. Springer, 2006.

[40] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang. High-speed high-security signatures. *Journal of Cryptographic Engineering*, 2(2), 2012.

[41] O. Berthold, H. Federrath, and S. Kopsell. Web MIXes: A system for anonymous and unobservable Internet access. In *Proceedings of the LNCS Workshop on Design Issues in Anonymity and Unobservability*, 2001.

[42] A. Bittau, D. B. Giffin, M. Handley, D. Mazières, Q. Slack, and E. W. Smith. Cryptographic protection of TCP Streams (tcpcrypt). draft-ietf-tcpinc-tcpcrypt-06, 2017.

[43] A. Bittau, M. Hamburg, M. Handley, D. Mazières, and D. Boneh. The Case for Ubiquitous Transport-level Encryption. In *Proceedings of the USENIX Security Symposium*, 2010.

[44] M. S. Blumenthal and D. D. Clark. Rethinking the Design of the Internet: The End-to-End Arguments vs. the Brave New World. *Internet Technology, ACM Transactions on*, 2001.

[45] N. Brownlee and K. Claffy. Understanding Internet Traffic Streams: Dragonflies and Tortoises. *Communication Magazine, IEEE*, 40(10), 2002.

[46] R. Bush and R. Austein. The Resource Public Key Infrastructure (RPKI) to Router Protocol. RFC 6810 (Proposed Standard), Jan. 2013.

[47] CAIDA. Observing Routing Asymmetry in Internet Traffic. https://goo.gl/uE4V3B.

[48] D. L. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, 24(2), 1981.

[49] C. Chen, D. E. Asoni, D. Barrera, G. Danezis, and A. Perrig. HORNET: High-speed Onion Routing at the Network Layer. In *Proceedings of the ACM Conference on Computer & Communications Security (CCS)*, 2015.

[50] Cisco. Cisco IOS Software Integrity Assurance. http://bit.ly/2ab76RE.

[51] D. D. Clark. *Designs for an Internet*. 2017.

[52] D. D. Clark, B. Lehr, S. Bauer, P. Faratin, R. Sami, and J. Wroclawski. Overlay Networks and the Future of the Internet. *Communications & Strategies*, (63), 2006.

[53] Communications Alliance Ltd. Internet Service Providers Voluntary Code of Practice: For Industry Self-Regulation in the Area of Cyber Security. Technical report, Communications Alliance Ltd, 2014.

[54] G. Danezis, R. Dingledine, and N. Mathewson. Mixminion: Design of Type III Anonymous Remailer Protocol. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2003.

[55] C. Díaz, S. Seys, J. Claessens, and B. Preneel. Towards Measuring Anonymity. In *Proceedings of the Workshop on Designing Privacy Enhancing Technologies (PETS)*, 2003.

[56] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-generation Onion Router. In *Proceedings of the USENIX Security Symposium*, 2004.

[57] Z. Duan, X. Yuan, and J. Chandrashekar. Constructing Inter-Domain Packet Filters to Control IP Spoofing Based on BGP Updates. In *Proceedings of IEEE INFOCOMM*, 2006.

[58] N. G. Duffield and M. Grossglauser. Trajectory Sampling for Direct Traffic Observation. *Networking, IEEE/ACM Transactions on*, 9(3), 2001.

[59] V. Dukhovni. Opportunistic Security: Some Protection Most of the Time. RFC 7435 (Informational), Dec. 2014.

[60] R. Durairajan, S. K. Mani, J. Sommer, and P. Barford. Time's Forgotten: Using NTP to Understand Internet Latency. In *Proceedings of the ACM Workshop on Hot Topics in Networks (HotNets)*, 2015.

[61] M. Dworkin. Recommendation for Block Cipher Modes of Operation: Methods for Format-Preserving Encryption. *NIST Special Publication*, 800, 2016.

[62] 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 *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2012.

[63] V. Fajardo, J. Arkko, J. Loughney, and G. Zorn. Diameter Base Protocol. RFC 6733 (Proposed Standard), Oct. 2012. Updated by RFC 7075.

[64] D. Farinacci, T. Li, S. Hanks, D. Meyer, and P. Traina. Generic Routing Encapsulation (GRE). RFC 2784 (Proposed Standard), Mar. 2000. Updated by RFC 2890.

[65] A. Farrel and S. Farrell. Opportunistic Security in MPLS Networks. draft-ietf-mpls-opportunistic-encrypt-03, 2017.

[66] S. Farrell and H. Tschofenig. Pervasive Monitoring Is an Attack. RFC 7258 (Best Current Practice), May 2014.

[67] S. K. Fayazbaksh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. M. Maggs, K. NG, V. Sekar, and S. Shenker. Less Pain, Most of the Gain: Incrementally Deployable ICN. In *Proceedings of the ACM Conference on SIGCOMM*, 2013.

[68] Federal Trade Commission. FTC Shuts Down Notorious Rogue Internet Service Provider. https://goo.gl/5jYFdV, June 2009.

[69] A. Feldmann. Internet Clean-Slate Design: What and Why? *Computer Communication Review, ACM SIGCOMM*, 2007.

[70] P. Felix, P. Sanders, and J. Singler. Cache-, Hash- and Space-efficient Bloom Filters. *Journal of Experimental Algorithmics (JEA)*, 14(4), 2009.

[71] E. Fink. Stalker: A creepy look at you, online. http://goo.gl/DWCiKI, Jun 2014.

[72] D. Fisher. A Look Behind the Future Internet Architectures Efforts. *Computer Communication Review, ACM SIGCOMM*, 44(3), 2014.

[73] S. Frankel, R. Glenn, and S. Kelly. The AES-CBC Cipher Algorithm and Its Use with IPsec. RFC 3602 (Proposed Standard), Sept. 2003.

[74] T. Gerbet, A. Kumar, and C. Lauradoux. The Power of Evil Choices in Bloom Filters. Technical Report hal-01082158, INRIA Grenoble, 2014.

[75] R. Grandl, D. Han, S.-B. Lee, H. Lim, M. Machado, M. K. Mukerjee, and D. Naylor. Supporting Network Evolution and Incremental Deployment with XIA. *Computer Communication Review, ACM SIGCOMM*, 42(4), 2012.

[76] S. Gueron. Intel Advanced Encryption Standard (AES) New Instruction Set. https://goo.gl/of08Dg, Mar 2010.

[77] C. Gulcu and G. Tsudik. Mixing E-mail with Babel. In *Proceedings of the Symposium on Network and Distributed System Security (NDSS)*, 1996.

[78] D. Han, A. Anand, F. Dogar, B. Li, H. Lim, M. Machado, A. Mukundan, W. Wu, A. Akella, D. G. Andersen, J. W. Byers, S. Seshan, and P. Steenkiste. XIA: Efficient Support for Evolvable Internetworking. In *Proceedings of the USENIX Conference on Networked Systems Design and Implementation (NSDI)*, 2012.

[79] S. Han, V. Liu, Q. Pu, S. Peter, T. Anderson, A. Krishnamurthy, and D. Wetherall. Expressive Privacy Control with Pseudonyms. In *Proceedings of the ACM Conference on SIGCOMM*, 2013.

[80] M. Handley. Why the Internet Just Works. *BT Technology Journal*, 24(3), 2006.

[81] S. Hanks, T. Li, D. Farinacci, and P. Traina. Generic Routing Encapsulation (GRE). RFC 1701 (Informational), Oct. 1994.

[82] J. Hayes and G. Danezis. k-fingerprinting: A Robust Scalable Website Fingerprinting Technique. In *Proceedings of the USENIX Security Symposium*, 2016.

[83] Y. He, M. Faloutsos, S. Krishnamurthy, and B. Huffaker. On Routing Asymmetry in the Internet. In *Proceedings of the IEEE Global Communications Conference (GLOBECOM)*, 2005.

[84] R. M. Hinden. Why Take Over the Hosts When You Can Take Over the Network. http://bit.ly/2apRAxZ, Feb 2014.

[85] H.-C. Hsiao, T. H.-J. Kim, S. B. Lee, X. Zhang, S. Yoo, V. D. Gligor, and A. Perrig. STRIDE:Sanctuary Trail—Refuge from Internet DDoS Entrapment. In *Proceedings of the ACM Asia Conference on Computer and Communications Security (AsiaCCS)*, 2013.

[86] N. Hu, L. E. Li, M. Z. Mao, P. Steenkiste, and J. Wang. Locating Internet Bottlenecks: Algorithms, Measurements, and Implications. In *Proceedings of the ACM Conference on SIGCOMM*, 2004.

[87] G. Huston, G. Michaelson, and R. Loomans. A Profile for X.509 PKIX Resource Certificates. RFC 6487 (Proposed Standard), Feb. 2012. Updated by RFC 7318.

[88] C. Imbrenda, L. Muscariello, and D. Rossi. Analyzing Cacheable Traffic in ISP Access Networks for Micro CDN Applications via Content-Centric Networking. In *Proceedings of the ACM Conference on Information-Centric Networking (ICN)*, 2014.

[89] ISO. Intermediate System to Intermediate System intra-domain routing information exchange protocol for use in conjunction with th protocol for providing the connectionless-mode network service (ISO 8473). *International Standard*, 10589, 2002.

[90] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard. Networking Named Content. In *Proceedings of the ACM Conference on Emerging Networking EXperiments and Technologies (CoNEXT)*, 2009.

[91] A. Jerichow, J. Muller, A. Pfitzmann, B. Pfitzmann, and M. Waidner. Real-Time Mixes: A Bandwidth-Efficient Anonymity Protocl. *Selected Areas in Communications, IEEE Journal on*, 1998.

[92] S. Jiang, D. Guo, and B. Carpenter. An Incremental Carrier-Grade NAT (CGN) for IPv6 Transition. RFC 6264 (Informational), June 2011.

[93] C. Jin, H. Wang, and K. G. Shin. Hop-count Filtering: An Effective Defense Against Spoofed DDoS Traffic. In *Proceedings of the ACM Conference on Computer & Communications Security (CCS)*, 2003.

[94] A. Johnson, C. Wacek, R. Jansen, M. Sherr, and P. Syverson. Users Get Routed: Traffic Correlation on Tor by Realistic Adversaries. In *Proceedings of the ACM Conference on Computer & Communications Security (CCS)*, 2013.

[95] M. Juarez, M. Imani, M. Perry, C. Diaz, and M. Wright. Toward an Efficient Website Fingerprinting Defense. In *Proceedings of the European Symposium on Research in Computer Security (ESORICS)*, 2016.

[96] M. S. Kang and V. D. Gligor. Routing Bottlenecks in the Internet—Causes, Exploits, and Countermeasures. In *Proceedings of the ACM Conference on Computer & Communications Security (CCS)*, 2014.

[97] M. S. Kang, S. B. Lee, and V. D. Gligor. The Crossfire Attack. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2013.

[98] E. Katz-Bassett, H. V. Madhyastha, V. K. Adhikari, C. Scott, J. Sherry, P. v. Wesep, T. Anderson, and A. Krishnamurthy. Reverse Traceroute. In *Proceedings of the USENIX Conference on Networked Systems Design and Implementation (NSDI)*, 2010.

[99] C. Kaufman, P. Hoffman, Y. Nir, P. Eronen, and T. Kivinen. Internet Key Exchange Protocol Version 2 (IKEv2). RFC 7296 (Internet Standard), Oct. 2014. Updated by RFCs 7427, 7670.

[100] K. Ken, H. Young, M. Luckie, and K. Claffy. Internet-scale IPv4 Alias Resolution with MIDAR. *Networking, IEEE/ACM Transactions on*, 21(2), 2013.

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

[102] S. Kent and K. Seo. Security Architecture for the Internet Protocol. RFC 4301 (Proposed Standard), Dec. 2005. Updated by RFCs 6040, 7619.

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

[104] J. Kirk. ISP Cut off From Internet After Security Concerns. https://goo.gl/QUbFwP, 2008.

[105] J. Kirk and R. McMillan. After weeklong fight, rogue ISP Troyak struggles for life. https://goo.gl/2Ccheu, 2010.

[106] B. Krishnamurthy and C. E. Wills. Privacy Diffusion on the Web: A Longitudinal Perspective. In *Proceedings of the ACM Conference on World Wide Web (WWW)*, 2009.

[107] S. Le Blond, D. Choffnes, W. Caldwell, P. Druschel, and N. Merrit. Herd: A Scalable, Traffic Analysis Resistant Anonymity Network for VoIP Systems. In *Proceedings of the ACM Conference on SIGCOMM*, 2015.

[108] S. Le Blond, D. Choffnes, W. Zhou, P. Druschel, H. Ballani, and P. Francis. Towards Efficient Traffic-analysis Resistant Anonymity Networks. In *Proceedings of the ACM Conference on SIGCOMM*, 2013.

[109] T. Lee and B. Hau. The New Route to Persistence: Compromised Routers in the Wild. http://bit.ly/1ObMm7u, Sep 2015.

[110] T. Lee, C. Pappas, A. Perrig, V. D. Gligor, and Y.-C. Hu. The Case for In-Network Replay Suppression. In *Proceedings of the ACM Asia Conference on Computer and Communications Security (AsiaCCS)*, 2017.

[111] M. Lepinski, S. Kent, and D. Kong. A Profile for Route Origin Authorizations (ROAs). RFC 6482 (Proposed Standard), Feb. 2012.

[112] J. Li, J. Mirkovic, M. Wang, P. Reiher, and L. Zhang. SAVE: source address validity enforcement protocol. In *Proceedings of IEEE INFOCOMM*, 2002.

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

[114] V. Liu, S. Han, A. Krishnamurthy, and T. Anderson. Tor Instead of IP. In *Proceedings of the ACM Workshop on Hot Topics in Networks (HotNets)*, 2011.

[115] X. Liu, A. Li, X. Yang, and D. Wetherall. Passport: Secure and Adoptable Source Authentication. In *Proceedings of the USENIX Conference on Networked Systems Design and Implementation (NSDI)*, 2008.

[116] X. Liu, X. Yang, and Y. Lu. To Filter or to Authorize: Network-Layer DoS Defense Against Multimillion-node Botnets. In *Proceedings of the ACM Conference on SIGCOMM*, 2008.

[117] X. Liu, X. Yang, D. Wetherall, and T. Anderson. Efficient and Secure Source Authentication with Packet Passports. In *Proceedings of the USENIX Workshop on Steps to Reducing Unwanted Traffic on the Internet (SRUTI)*, 2006.

[118] J. Livingood, N. Mody, and M. O'Reirdan. Recommendations for the Remediation of Bots in ISP Networks. RFC 6561 (Informational), Mar. 2012.

[119] T. A. Longstaff, J. T. Ellis, S. V. Hernan, H. F. Lipson, R. D. McMillan, L. H. Pesante, and D. Simmel. Security of the Internet. *The Froehlich/Kent Encyclopedia of Telecommunications*, 15:231–255, 1997.

[120] S.-W. Luan and V. D. Gligor. On Replay Detection in Distributed Systems. In *Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS)*, 1990.

[121] m3aawg. ABCs for ISPs. https://www.m3aawg.org/abcs-for-ISP-code, 2017.

[122] 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*, 297, 2009.

[123] H. Marouani and M. R. Dagenais. Comparing High Resolution Timestamps in Computer Clusters. In *Proceedings of the IEEE Canadian Conference on Electrical and Computer Engineering*, 2005.

[124] D. McGrew and J. Viega. The Use of Galois Message Authentication Code (GMAC) in IPsec ESP and AH. RFC 4543 (Proposed Standard), May 2006.

[125] D. A. McGrew and J. Viega. The Galois/Counter Mode of Operation (GCM). http://goo.gl/9sl9kK, 2004.

[126] S. W. L. Meiser. *Quantitative Anonymity Guarantees for Tor*. PhD thesis, Saarland University, 2016.

[127] A. J. Menezes, P. C. V. Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 2001.

[128] U. Moeller, L. Cottrell, P. Palfrader, and L. Sassaman. Mixmaster Protocol Version 2. draft-sassaman-mixmaster-03.txt, 2005.

[129] A. Morton. IMIX Genome: Specification of Variable Packet Sizes for Additional Testing. RFC 6985 (Informational), July 2013.

[130] J. Moy. OSPF Version 2. RFC 2328 (Internet Standard), Apr. 1998. Updated by RFCs 5709, 6549, 6845, 6860, 7474, 8042.

[131] M. K. Mukerjee, D. Han, S. Seshan, and P. Steenkiste. Understanding Tradeoffs in Incremental Deployment of New Network Architectures. In *Proceedings of the ACM Conference on Emerging Networking EXperiments and Technologies (CoNEXT)*, 2013.

[132] J. Naous, M. Walfish, A. Nicolosi, D. Mazières, M. Miller, and A. Seehra. Verifying and enforcing network paths with ICING. In *Proceedings of the ACM Conference on Emerging Networking EXperiments and Technologies (CoNEXT)*, 2011.

[133] T. Narten, R. Draves, and S. Krishnan. Privacy Extensions for Stateless Address Autoconfiguration in IPv6. RFC 4941 (Draft Standard), Sept. 2007.

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

[135] M. Nottingham and M. Thomson. Opportunistic Security for HTTP/2. RFC 8164 (Experimental), May 2017.

[136] NSF. NSF Future Internet Architecture Project. www.nets-fia.net.

[137] OECD. THE ROLE OF INTERNET INTERMEDIARIES IN ADVANCING PUBLIC POLICY OBJECTIVES. Technical Report JT03304378, Organisation for Economic Co-operation and Development (OECD), 2011.

[138] C. H. Papadimitriou. On the Complexity of Integer Programming. *Journal of the ACM (JACM)*, 28(4), 1981.

[139] C. Pappas, R. M. Reischuk, and A. Perrig. FAIR:Forwarding Accountability for Internet Reputability. In *Proceedings of the IEEE Conference on Network Protocols (ICNP)*, 2015.

[140] K. Park and H. Lee. On the Effectiveness of Route-Based Packet Filtering for Distributed DoS Attack Prevention in Power-Law Internets. In *Proceedings of the ACM Conference on SIGCOMM*, 2001.

[141] A. Perrig, P. Szalachowski, R. M. Reischuk, and L. Chuat. *SCION: A Secure Internet Architecture*. Springer International Publishing, 2017.

[142] V. C. Perta, M. V. Barbera, G. Tyson, H. Haddadi, and A. Mei. A Glance through the VPN Looking Glass: IPv6 Leakage and DNS Hijacking in Commercial VPN Clients. In *Proceedings on Privacy Enhancing Technologies (PoPETs)*, 2015.

[143] L. Peterson, T. Anderson, D. Culler, and T. Roscoe. A Blueprint for Introducing Disruptive Technology into the Internet. *Computer Communication Review, ACM SIGCOMM*, 33(1), 2003.

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

[145] A. Pfitzmann and M. Hansen. Anonymity, Unobservability, and Pseudonymity: A Consolidated Proposal for Terminology. http://goo.gl/n4tciX, Jul 2000.

[146] A. Pfitzmann, B. Pfitzmann, and M. Waidner. ISDN-mixes: Untraceable communication with very small bandwidth overhead. In *Proceedings of the GI/ITG Conference on Communication in Distributed Systems*, 1991.

[147] P. Porras. Towards a More Secure SDN Control Layer - SRI International's View. http://bit.ly/2ax0ERr, Oct 2013.

[148] QUIC. QUIC, a multiplexed stream transport over UDP. www.chromium.org/quic.

[149] B. Raghavan, T. Kohno, A. C. Snoeren, and W. David. Enlisting ISPs to improve online privacy: IP Address Mixing by Default. In *Proceedings of the Privacy Enhancing Technologies Symposium (PETS)*, 2009.

[150] B. Raghavan and A. C. Snoeren. A System for Authenticated Policy-Compliant Routing. In *Proceedings of the ACM Conference on SIGCOMM*, 2004.

[151] M. G. Reed, P. F. Syverson, and D. M. Goldschlag. Anonymous Connections and Onion Routing. *Selected Areas in Communications, IEEE Journal on*, 16(4), 1998.

[152] J. Rexford and C. Dovrolis. Future Internet Architecture: Clean-Slate Versus Evolutionary Research. *Communications of the ACM*, 2010.

[153] M. Richardson and D. Redelmeier. Opportunistic Encryption using the Internet Key Exchange (IKE). RFC 4322 (Informational), Dec. 2005.

[154] C. Rigney, S. Willens, A. Rubens, and W. Simpson. Remote Authentication Dial In User Service (RADIUS). RFC 2865 (Draft Standard), June 2000. Updated by RFCs 2868, 3575, 5080, 6929, 8044.

[155] F. Roesner, T. Kohno, and D. Wetherall. Detecting and Defending Against Third-Party Tracking on the Web. In *Proceedings of the USENIX Conference on Networked Systems Design and Implementation (NSDI)*, 2012.

[156] B. Rowe, D. Wood, D. Reeves, and F. Braun. The Role of Internet Service Providers in Cyber Security. http://goo.gl/d1Ky3a, 2011.

[157] A. Sakurai, T. Minohara, R. Sato, and K. Mizutani. One-Time Receiver Address in IPv6 for Protecting Unlinkability. In *Proceedings of the Asian Computing Science Conference (ASIAN)*, 2007.

[158] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-End Arguments in System Design. *Computer Systems, ACM Transactions on*, 2(4), 1984.

[159] J. Sankey and M. Wright. Dovetail: Stronger Anonymity in Next-Generation Internet Routing. In *Proceedings of the Privacy Enhancing Technologies Symposium (PETS)*, 2014.

[160] A. Serjantov and G. Danezis. Towards an Information Theoretic Metric for Anonymity. In *Proceedings of the Workshop on Designing Privacy Enhancing Technologies (PETS)*, 2003.

[161] M. Shaw. Leveraging Good Intentions to Reduce Unwanted Network Traffic. In *Proceedings of the USENIX Workshop on Steps to Reducing Unwanted Traffic on the Internet (SRUTI)*, 2006.

[162] V. Shmatikov and M.-H. Wang. Timing Analysis in Low-latency Mix Networks: Attacks and Defenses. In *Proceedings of the European Symposium on Research in Computer Security (ESORICS)*, 2006.

[163] W. Simpson. IP in IP Tunneling. RFC 1853 (Informational), Oct. 1995.

[164] I. Society. State of IPv6 Deployment 2017. Technical report, Internet Society, 2017.

[165] N. Spring, R. Mahajan, and D. Wetherall. Measuring ISP Topologies with Rocketfuel. In *Proceedings of the ACM Conference on SIGCOMM*, 2002.

[166] R. Stewart. Stream Control Transmission Protocol. RFC 4960 (Proposed Standard), Sept. 2007. Updated by RFCs 6096, 6335, 7053.

[167] A. Studer and A. Perrig. The Coremelt Attack. In *Proceedings of the European Symposium on Research in Computer Security (ESORICS)*, 2009.

[168] Y. Sun, A. Edmundson, L. Vanbever, O. Li, J. Rexford, M. Chiang, and P. Mittal. RAPTOR: Routing Attacks on Privacy in Tor. In *Proceedings of the USENIX Security Symposium*, 2015.

[169] P. Syverson, G. Tsudik, M. Reed, and C. Landwehr. Towards an Analysis of Onion Routing Security. In *Proceedings of the Workshop on Designing Privacy Enhancing Technologies (PETS)*, 2001.

[170] J. Viega and D. McGrew. The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP). RFC 4106 (Proposed Standard), June 2005.

[171] X. Wang, S. Chen, and S. Jajodia. Tracking Anonymous Peer-to-Peer VoIP Calls on the Internet. In *Proceedings of the ACM Conference on Computer & Communications Security (CCS)*, 2005.

[172] A. F. Westin. *Privacy and Freedom*. Scribner, 1967.

[173] M. Wuergler. Secrets in Your Pocket. https://goo.gl/J16Ykx.

[174] A. Yaar, A. Perrig, and D. Song. SIFF: A Stateless Internet Flow Filter to Mitigate DDoS Flooding Attacks. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2004.

[175] I. Yamagata, Y. Shirasaki, A. Nakagawa, J. Yamaguchi, and H. Ashida. NAT444. https://tools.ietf.org/id/draft-shirasaki-nat444-06.txt, 2012.

[176] X. Yang, D. Wetherall, and T. Anderson. A Dos-limiting Network Architecture. In *Proceedings of the ACM Conference on SIGCOMM*, 2005.

[177] K. Zetter. NSA Laughs at PCs, Prefers Hacking Routers and Switches. http://bit.ly/2awxsN8.

[178] X. Zhang, H.-C. Hsiao, G. Hasker, H. Chan, A. Perrig, and D. G. Andersen. SCION: Scalability, Control, and Isolation on Next-Generation Networks. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2011.

[179] Y. Zhu, X. Fu, B. Graham, R. Bettati, and W. Zhao. On Flow Correlation Attacks and Countermeasures in Mix Networks. In *Proceedings of the Workshop on Designing Privacy Enhancing Technologies (PETS)*, 2004.

Curriculum Vitae: Tae-Ho Lee

Education

2013–2017 **Ph.D., Computer Science**
ETH Zurich, Switzerland
2007–2009 **M.S., Electrical Engineering**
Stanford University, USA
2004–2007 **B.S., Electrical and Computer Engineering**
University of Texas at Austin, USA

Experience

2013–2017 **Research Assistant**
Institute of Information Security, ETH Zurich, Switzerland
2009–2013 **Research Staff**
Electronics and Telecommunications Research Institute, S.Korea
2008–2008 **Research Assistant**
Computer Graphics Lab, Stanford University, USA
2007–2007 **Research Assistant**
Transceiver Technology Lab, Seoul National University, S.Korea
2005–2005 **Intern**
KT Corporation, S.Korea

Teaching Experience

2015–2017 Current Topics in Information Security
2016–2017 Network Security
2014–2017 Operating Systems and Networks
2006–2007 Digital Logic Design

Publications

[1] Taeho Lee, Christos Pappas, Adrian Perrig, "Bootstrapping Privacy in Today's Internet," *under submission*, 2017.

[2] Taeho Lee*, Christos Pappas*, Pawel Szalachowski*, Adrian Perrig, "Towards Sustainable Evolution for the TLS Public-Key Infrastructure," *under submission*, 2017.

[3] Taeho Lee*,Christos Pappas*, Adrian Perrig, Virgil Gligor, Yih-Chun Hu, "The Case for In-Network Reply Suppression," *In Proceedings of the ACM Asia Conference on Computer and Communications Security (AsiaCCS)*, 2017.

[4] Taeho Lee, Christos Pappas, David Barrera, Pawel Szalachowski, Adrian Perrig, "Source Accountability with Domain-brokered Privacy," *In Proceedings of the ACM Conference on Emerging Networking Experiments and Technologies (CoNEXT)*, 2016.

[5] Taeho Lee, Christos Pappas, Pawel Szalachowski, Adrian Perrig, "Communication Based on Per-Packet One-Time Addresses," *In Proceedings of the IEEE Conference on Network Protocols (ICNP)*, 2016.

[6] Takayuki Sasaki, Christos Pappas, Taeho Lee, Torsten Hoefler, Adrian Perrig, "SDNsec: Forwarding Accountability for the SDN Data Plane," *In Proceedings of the IEEE International Conference on Computer Communication and Networks (ICCCN)*, 2016.

[7] Pawel Szalachowski, Laurent Chuat, Taeho Lee, Adrian Perrig, "RITM: Revocation in the Middle," *In Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2016.

[8] Yi-Hsuan Kung, Taeho Lee, Po-Ning Tseng, Hsu-Chun Hsiao, Tiffany Hyun-Jin Kim, Soo Bum Lee, Yue-Hsun Lin, Adrian Perrig, "A Practical System for Guaranteed Access in the Presence of DDoS Attacks and Flash Crowds," *In Proceedings of the IEEE Conference on Network Protocols (ICNP)*, 2015.

[9] Taeho Lee*, Christos Pappas*, Cristina Basescu, Jun Han, Torsten Hoefler, Adrian Perrig, "Source-Based Path Selection: The Data Plane Perspective," *In Proceedings of the ACM Conference on Future Internet Technologies (CFI)*, 2015.

[10] Taesang Choi, Taeho Lee, Nodir Kodirov, Jaegi Lee, Doyeon Kim, Joon-Myung Kang, Sungsu Kim, John Strassner, James Won-Ki Hong, "HiMang: Highly Manageable Network and Service Architecture for New Generation," *Journal of Communications and Networks*, Vol. 13, pp.552-566, 2012.

[11] Woojik Chun, Taeho Lee, Taesang Choi, "YANAIL: Yet Another definition on Name, Addresses, Identifiers, and Locators," *In Proceedings of the ACM Conference on Future Internet Technologies (CFI)*, 2011.

[12] Sungkee Noh, Euisin Lee, Seungmin Oh, Taeho Lee, Sang-Ha Kim, "Effective retransmission scheme for supporting Communication Reliability in Sensor Networks," *In Proceedings of the IEEE Symposium on Personal Indoor and Mobile Radio Communications (PIMRC)*, 2010.