Communication Based on Per-Packet One-Time Addresses

Taeho Lee
ETH Zürich
kthlee@inf.ethz.ch

Christos Pappas
ETH Zürich
pappasch@inf.ethz.ch

Pawel Szalachowski
ETH Zürich
psz@inf.ethz.ch

Adrian Perrig
ETH Zürich
aperrig@inf.ethz.ch

***Abstract*—The act of communication on the Internet inevitably leaks information. In particular, network headers reveal information (e.g., source address, flow information); yet, protecting the header has proven challenging. Past research successfully protected certain fields of the headers (e.g., source address), but no proposal has attempted to eliminate flow information from the header so that packets cannot be linked to flows; flow information is systematically used to subvert privacy. Hence, we investigate the following questions: Can we design an architecture that eliminates flow-packet linkability? Can we do so without imposing impractical requirements on the network infrastructure?**

Our proposed architecture is based on per-packet One Time Address (OTA)—an address that a host uses to send or receive exactly one packet. Furthermore, the architecture eliminates any implicit (e.g., the standard five-tuple in TCP/UDP packets) or explicit (e.g., flow identifier) flow information from packet headers. Yet, the architecture allows the communicating hosts to demultiplex seemingly unrelated packets to flows. We have implemented the proposed architecture, and our evaluation shows that it can satisfy today's packet forwarding requirements.

I. Introduction

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 [1–4].

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 [5]. APIP completely removes the "source address" from the network header in order to eliminate common source information across flows [6].

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 [7–11]. 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 paper, 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.

Our approach to achieve flow-packet unlinkability is communication based on (disposable) *One-Time Addresses* (OTAs): once an address has been used in the network header, it is not re-used in subsequent packets. This means that source addresses are not even re-used as destination addresses in subsequent response packets.

Furthermore, we propose pervasive encryption with perfect forward secrecy for all packet payloads. This step is necessary to prevent leaked information 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.

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

- flow demultiplexing at the end hosts
- key management for pervasive network-layer encryption
- efficient OTA management
  - secure and optimized OTA generation
  - no per-OTA state in the network infrastructure.

Furthermore, we present a software-router prototype that can forward up to 20M packets per second, and that can saturate a capacity of 110 Gbps even on a commodity desktop PC.

II. Problem Setup

A. 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 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) [12]: even if an adversary obtains all long-term keys of a host, he cannot subvert data privacy of past communication.

B. 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 VII for more detail.). We also assume that the cryptographic primitives we use are secure: signatures cannot be forged and encryptions cannot be broken.

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

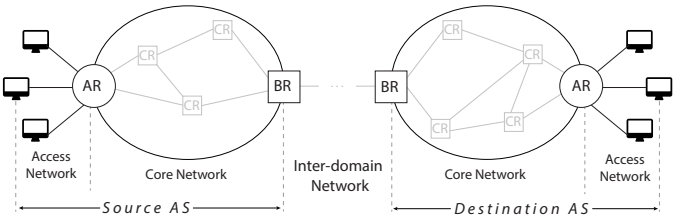


Fig. 1: The infrastructure of an AS and the 3-layer structure of the data plane that supports OTA-based communication.

III. 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 from the packet header. 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.

A. 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.

B. 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 VI-A) 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 communication, 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 [13]. In Section VII, we show how to enlarge the anonymity set for small ISPs.

C. Architectural Components

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

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
$OTA^{\langle N \rangle}_{H_i}$	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 IV-D)
k_{AS_i}	Symmetric key known among the infrastructure (e.g., routers) within AS_i
$E_k(\bullet)/E^{-1}_k(\bullet)$	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 I: Summary of Symbols and Notation.

in the access network, packet headers contain the uniquely assigned HIDs of the hosts. We discuss the security implications of our approach in Section VI-B.

IV. 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 I summarizes the notation that we use throughout the section.

A. Assumptions

- Every AS has a public key and a corresponding certificate; and there is a public-key infrastructure (e.g., RPKI [14]) from which an entity can retrieve and verify AS-certificates.
- Communication in the access network is secure, i.e., packet payloads are encrypted. For example, hosts can establish IPsec sessions with their ARs.

B. 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 1).

$$OTA = E_{k_{AS}}(HID, FID) \quad (1)$$

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, note that 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 V.

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 IV-D) 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.

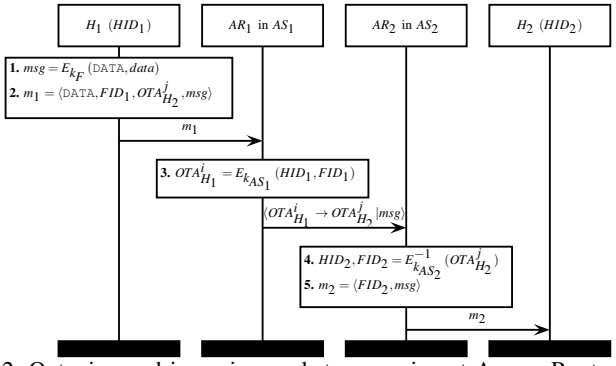


Fig. 2: Outgoing and incoming packet processing at Access Routers

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 IV-E.

C. 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 precomputed OTA encodes FID_2 that is used by H_2 to identify the flow. ARs process outgoing and incoming packets differently (Figure 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 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.

When AR_2 receives an incoming packet from the core network, it performs the following tasks (Line 4-5): 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 4).

D. 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 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 and H_2 have received OTAs ($OTA^1_{H_1}$ and $OTA^2_{H_2}$, respectively) and certificates ($C_{OTA^1_{H_1}}$ and $C_{OTA^2_{H_2}}$, respectively) from their respective ASes. These OTAs are used as the source addresses for connection establishment packets. Second, we assume that H_1 has obtained an OTA for H_2 ($OTA^1_{H_2}$) and the associated certificate ($C_{OTA^1_{H_2}}$), 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 packet of the connection establishment (Line 1). The symmetric key is supplied to AR_1 to encrypt a second OTA of the source ($OTA^2_{H_1}$)—the

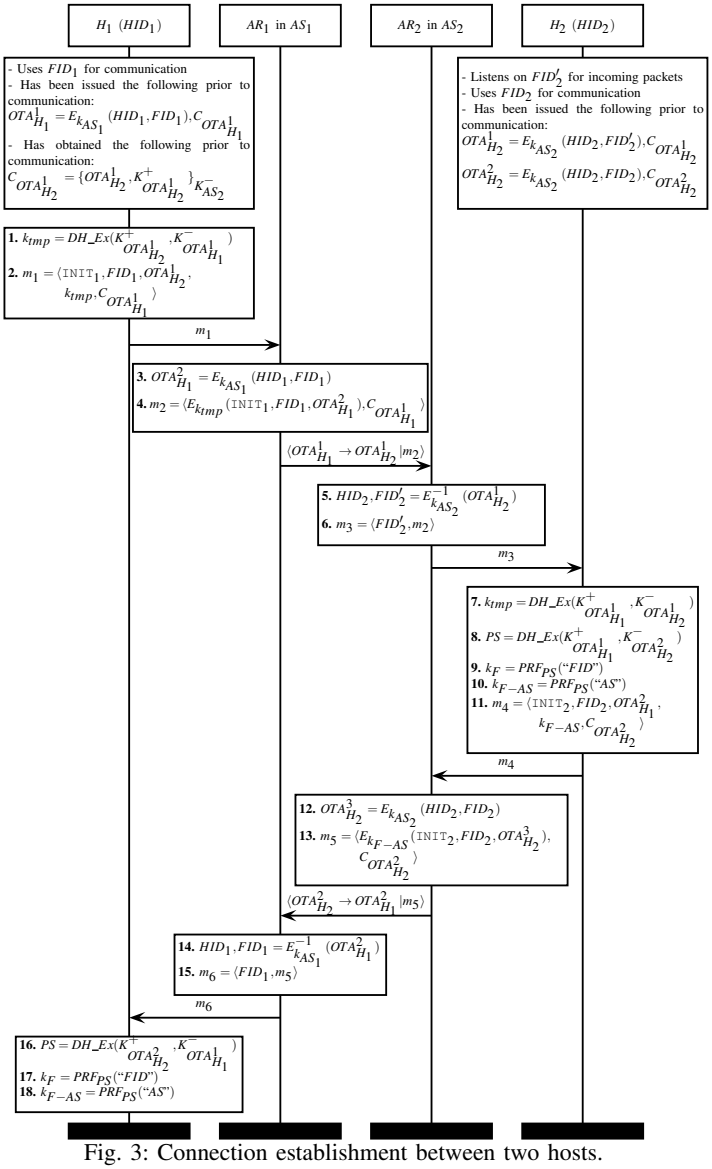


Fig. 3: Connection establishment between two hosts.

OTA that will be used in H_2's reply packet back to the source. Then H_1 constructs a special connection-establishment message ($\texttt{INIT}_1$) for its AR. The message contains the FID_1, the destination's OTA ($OTA^1_{H_2}$), the symmetric key k_{tmp}, and the certificate of $OTA^1_{H_1}$ whose public key will be used to generate symmetric keys for data encryption.

AR_1 identifies that the incoming packet is used for connection establishment. First, AR_1 generates an OTA that will be used as the reply address (replyOTA) from H_2. Then, AR_1 constructs a message for H_2 (Line 4), which includes the reply address $OTA^2_{H_1}$, encrypted with the symmetric key k_{tmp}. Additionally, AR_1 includes the certificate ($C_{OTA^1_{H_1}}$) of the OTA ($OTA^1_{H_1}$) that is used as the source address. This certificate is to prevent a Man-in-the-Middle (MitM) attack by AS_2, which would compromise data privacy between the two hosts. AR_1 sends the generated packet through the core network towards H_2. Then AR_2 finally forwards the packet to H_2 (Figure 2).

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 m_2. Then, H_2 computes a pre-shared secret (PS) using the public 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 VI-B).

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.

H_2 constructs a special connection-establishment message ($\texttt{INIT}_2$) for his AR; the message contains the FID_2, the destination's OTA ($OTA^2_{H_1}$), the symmetric key (k_{F-AS}), and the certificate for the source OTA ($C_{OTA^2_{H_2}}$).

AR_2 identifies that the incoming packet is used for connection establishment. Then, it generates an OTA to be used as a reply address by H_1 ($OTA^3_{H_2}$); the reply address is encrypted with the symmetric key k_{F-AS}. AR_2 sends the generated packet through the core network towards H_1. AR_1 intercepts the packet and forwards it as specified in Figure 2.

Finally, H_1 generates the symmetric keys k_F and k_{F-AS} using the public key in $C_{OTA^2_{H_2}}$. Then, it obtains $OTA^3_{H_2}$ by decrypting m_5 using k_{F-AS}.

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 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 (Line 1-2). Then, H_1 is forwarded to H_2 in the same way as data packets (Figure 2).

H_2 creates a reply packet (`OTA_REP`), which will be processed by AR_2 (Line 6). 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 8), encrypts them (Line 9), 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.

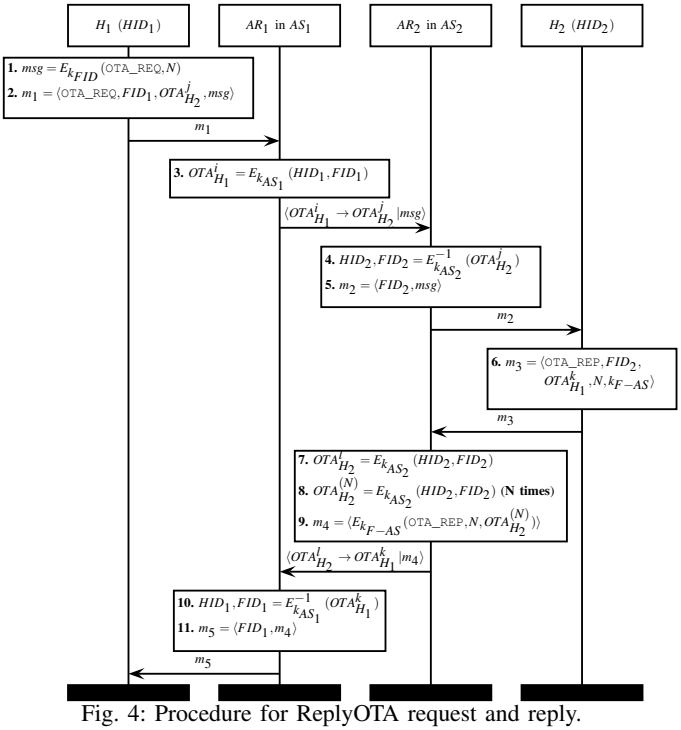


Fig. 4: Procedure for ReplyOTA request and reply.

E. 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 (Line 4 and Line 13 in Figure 3) so that both hosts know each other's FIDs.

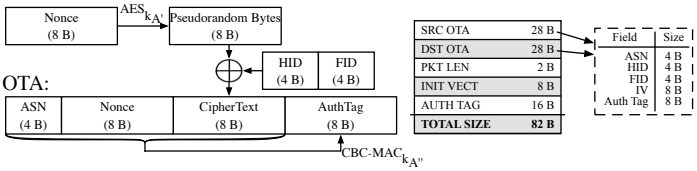
Certificate Issuance. The majority of OTAs are generated on the fly during end-to-end communication. However, a few OTAs must be generated proactively as the associated certificates are used for authentication and key negotiation during connection establishment.

We describe a certificate-issuance procedure that is used to generate OTA-certificatesThe 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.

V. Implementation

We describe the implementation of our main components.

One-Time Address. An OTA contains a host identifier (HID) and a flow identifier (FID), as Equation 1 specifies. We use 4 bytes for the HID, similar to IPv4. We also use 4 bytes for the



(a) OTA specification (b) Header specification.

Fig. 5: Specifications.

FID, which are sufficient to uniquely identify all concurrent flows that a host would maintain at any given time.

We need a CCA-secure OTA generation procedure, as mentioned in Section IV-B. To this end, we use authenticated encryption and specifically the "Encrypt-then-MAC" approach [15]. An OTA is computed as follows: first, a nonce is encrypted to generate random bits, which are then used to encrypt the HID and FID (CipherText). The nonce is a random value that can be used only once (for a given key) and ensures that a new OTA is generated even for the same (HID,FID) pair. Then, an authentication tag (AuthTag) is computed using the CipherText, nonce, and AS information as the input. We use a CBC-MAC to compute the AuthTag. Note that two different keys are used for encryption (k'_{AS}) and authentication (k''_{AS}); and they are derived from the AS key, k_{AS}.

OTA generation and decryption must be efficient since these operations are performed for every packet. Therefore, we have engineered the OTA structure for AES processing, since AES operations have widespread hardware support. That is, the symmetric encryption and CBC-MAC are both based on AES.

Packet Header. Figure 5b shows the packet header that we use in our implementation. The header contains the source and destination OTAs and the packet length that indicates the size of the packet. The replyOTA is not visible in the header; instead it is encrypted in the payload.

We use authenticated encryption to encrypt the payload between the communicating hosts and therefore the header contains an initialization vector and an authentication tag. Similar to IPSec [16], we use 8 bytes and 16 bytes for these fields. The total size of the header is 82 bytes.

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 IV-E). For the public-key operations, we use the ed25519 signature scheme [17] and the ed25519 SUPERCOP REF10 implementation [18] for its high performance and short key size (32 bytes) and signatures (64 bytes).

Access Network. For our implementation, we assume that the communication between the host and the access router is based on IP, and that IPSec is used to secure the communication. Among the supported encryption schemes in IPSec [16, 19], we use Galois/Counter Mode (GCM) based on AES because of its efficiency. More specifically, we use the AES-GCM implementation in the OpenSSL library since it takes advantage of AES-NI to accelerate encryption and decryption operations.

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 IV-D). In addition, K^+ becomes part of the certificate that is issued by the host's AS. We use curve25519 [20] to generate DH value pairs and use elliptic curve Diffie-Hellman (ECDH) for symmetric key negotiation.

VI. Evaluation

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

A. 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 [21], 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] Using the minimum packet size (64 bytes), i.e., the highest possible packet rate, we compute a packet rate of 6,000 packets-per-second (pps) by an individual host and an aggregate packet rate of $9 \cdot 10^6$ pps by 1,500 hosts.
- **Flow Generation Rate.** We use the CAIDA Anonymized Internet Traces Dataset [22] 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 over-estimate 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 across all our measurements:

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

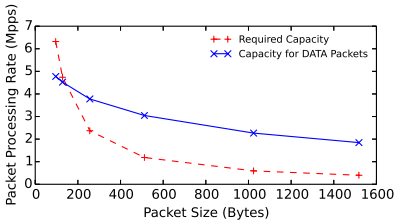


Fig. 6: Data packet processing rate by an Access Router.

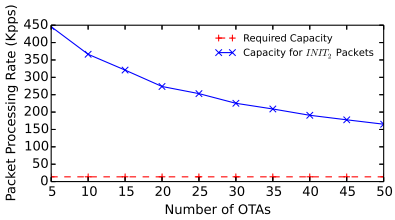


Fig. 7: Communication establishment packet processing rate by an Access Router.

- Communication in the access network is encrypted using IPSec based on AES-GCM. The AR maintains shared keys for each of the hosts in the access network; in total 1,500 symmetric keys are stored. We create a hash 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.

Q1: OTA and Certificate Generation. We evaluate the efficiency of OTA (Figure 5a) 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 IV-C, we described data packet forwarding at ARs. Specifically, for outgoing packets to the core network, the AR: 1) decrypts the message using the shared key with the sending host since the communication in the access network is encrypted, and 2) generates a OTA using the host's HID and FID in the message (m_1 in Figure 2). For incoming packets from the core network, the AR: 1) decrypts the destination OTA in the packet to obtain the HID and FID information, and 2) encrypts the FID and the payload of the incoming packet using the shared key with the destination host.

We measure the forwarding rate of an AR for six packet sizes: 96 bytes,[3] 128 bytes, 256 bytes, 512 bytes, 1024 bytes, and 1518 bytes. We run one million tests for each packet size and report the average.

Figure 6 shows the packet processing performance only for outgoing packets; the performance is the same for incoming packets, since the computation overhead of symmetric encryption and decryption are similar. The x-axis shows the packet size and the y-axis shows the processing performance in Million-packets-per-second (Mpps). The red-dotted line shows the number of packets that would be generated in an access network with 1,500 hosts that have a monthly data allowance of 1 TB. The solid blue line shows the packet processing rate by the AR. The result shows that for all but the two smallest packet sizes, the AR satisfies the processing requirement.

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 million end-hosts in an AS. That is, the CR contains a routing table with 2 million entries, where each entry consists of an HID and an outgoing port number as the key and the corresponding value, respectively. We report the average forwarding performance of a CR forwarding one million packets. Our evaluation shows that a CR forwards 96-byte and 1518-byte packets at 20.3 Mpps (15.6 Gbps) and 9.1 Mpps (110 Gbps), respectively. The performance results demonstrate that our approach is lightweight and can be implemented even on commodity machines.

Q3: Connection Establishment Processing. ARs perform different tasks during connection establishment: The AR that serves the initiating host, processes $\texttt{INIT}_1$ messages (Lines 3-4, Figure 3), and the AR that serves the listening host processes $\texttt{INIT}_2$ messages (Lines 12-13, Figure 3). We only evaluate $\texttt{INIT}_2$ messages since the processing overhead for $\texttt{INIT}_2$ messages is strictly greater than that for $\texttt{INIT}_1$ messages.

We generate 10^6 packets and report the average. Furthermore, we consider the case that the listening host replies to the initiating one with a number of N replyOTAs; we vary N from 5 to 50 in increments of 5 OTAs.

Figure 7 shows the packet processing performance by an AR. The red dotted line shows the peak flow rate that we observed in the Equinix-Chicago monitor; and the blue solid line with 'x' markers shows the packet processing rate for $\texttt{INIT}_2$ packets. The result shows a decrease in packet processing rate as the number of replyOTAs increase. In addition, the result shows that ARs can process connection establishment packets at a higher rate than the observed maximum flow rate on a backbone link.

B. 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

[3]96 bytes is the smallest packet size in OTA; a 96-byte packet consists of an Ethernet header (14 bytes) and a OTA header (82 bytes).

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.

VII. Discussion

Traffic Engineering. 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. Specifically, we envision an architecture that enables hosts to decide on the performance-privacy trade-off,[4] i.e., an architecture with tunable privacy properties.

Architecture with Tunable Privacy. We envision an architecture that provides privacy at two different layers: the network layer and the transport layer. The *network layer* should provide the building blocks to achieve basic privacy properties. Specifically, the architecture should enable users to decide how the network assigns their addresses: applications with strict privacy requirements could use OTAs; applications with stricter performance requirements could use per-flow addresses in order to minimize packet reordering. Furthermore, the architecture should facilitate key management for ubiquitous encryption: applications with higher privacy requirements can encrypt all data above the network header.

The *transport layer* should provide resilience against more sophisticated attacks (e.g., side-channel attacks). The application can decide on a "secure transport" protocol to use, according to its needs. For example, a privacy-sensitive application may use a transport protocol that pads all packets to a constant size and/or changes inter-packet timings to protect from side-channel attacks; an application without strict privacy requirements can fall back to traditional TCP.

Enlarging Anonymity Set for Source Hosts. Source OTAs are never used when routing packets, yet they identify a host at the granularity of its AS; the anonymity set becomes the size of the AS. The reason to reveal the source OTA in the packet is to support network troubleshooting: an entity in the network (e.g., router) may send a message (e.g., ICMP) back to the source.

We consider the following challenge: how can we hide the source OTA without sacrificing network troubleshooting? To this end, we define a third party that is responsible for identifying source ASes from source OTAs and forwarding ICMP messages to the source ASes. Then, we re-design the source OTA so that only the third party can correctly identify the source AS from source OTA.

In our design, we assume that each AS shares a symmetric key with the third party. Then, the source OTAs are modified as follows. First, the address of the third party is used in place of the source ASN. Second, we split the AuthTag in OTAs into two 4-byte MACs, MAC_1 and MAC_2, which are computed in an onion-style. MAC_1 is computed identically as the AuthTag in Figure 5a, but only the first four bytes are used. Then, MAC_2 is computed using the inputs of MAC_1 and MAC_1 itself as the inputs; the shared symmetric key with the third party is used for the MAC computation. Note that the source OTAs still contain the HID and FID information, since the source AS needs to deliver the ICMP message to the correct host.

Incremental Deployability. Communication based on OTAs can be used between a pair of deploying ASes, regardless of the deployment by transit ASes. Furthermore, a deploying AS does not need to completely modify its network—the AS needs to install *gateway(s)* and modify its access routers that serve the customers that are interested in using our architecture. Deploying ASes interpret the fields in OTAs as follows: they place the IPv4 addresses of the gateways in the ASN field and place the IPv4 addresses of the hosts in the HID field.

Then, communication based on OTAs is realized using a series of three IPv4 tunnels: a tunnel between an AR and a gateway in the source AS, a tunnel between gateways in the source and destination ASes, and a tunnel between the gateway and an AR in the destination AS. For the first and second tunnels, the source AR and gateway determine the addresses for the other end-points of the tunnel using the ASN information in source and destination OTAs, respectively. For the third tunnel, the destination gateway determines the address of the destination host by decrypting the destination OTA to obtain the destination HID.

[4] We are not the first to make this claim [23].

This deployment approach has a privacy implication—within an AS, the address of the host is visible as the IP tunnel headers contain hosts' addresses; hence, it is not possible to guarantee flow-packet unlinkability against an adversary who observes packets within the AS. However, as host addresses are not visible in the second tunnel (where a packet is forwarded across multiple ASes), an adversary that can eavesdrop on packets anywhere but within the source and destination ASes cannot link packets to flows. Moreover, once an AS fully deploys our architecture (i.e., all routers forward packets based on OTAs), this privacy implication disappears.

VIII. Related Work

Han et al. propose a pseudonym-based architecture that provides sender-application unlinkability [24]. Each user is provided with a pool of pseudonyms and can then decide how to use them: a single pseudonym for all applications or a separate pseudonym for each application. Similar to our approach, pseudonyms are encrypted tokens using the AS's key. However, the achieved privacy property is weaker than flow-packet unlinkability. Furthermore, the proposal does not specify the security properties of a pseudonym (e.g., CCA-security), which may leave space for an adversary to obtain the corresponding plaintexts, i.e., HIDs.

Raghavan et al. propose AS-wide NATs to mask the identifies of individual hosts [5]. APIP proposes the removal of the source address in the packet headers to prevent sender-flow correlation [6]. Both proposals provide sender-flow unlinkability. However, flow-packet unlinkability cannot be achieved due to flow-identifying information across packets. Furthermore, APIP claims data privacy, but does not describe key management, which by itself is a challenging problem.

Tcpcrypt [25] and MinimaLT [26] focus on pervasive encryption; however, both proposals fall short of providing sender-flow unlinkability. In Tcpcrypt only the TCP payload is encrypted. In MinimaLT, a tunnel identifier in packet headers leaks information: two packets with two different tunnel identifiers do not belong to the same sender.

IX. Conclusion

Designing an architecture that supports flow-packet unlinkability comes with many challenges. We have taken on these challenges and have proposed an architecture based on per-packet One Time Addresses (OTAs) that are used only once to send or receive packets. OTAs are designed so that ASes can efficiently generate them and do not need to maintain per-OTA state, and such that routers can efficiently process OTAs to meet today's packet forwarding requirements.

Future Work. The current Internet is too rigid to support diverse privacy requirements of the users. This paper shows that it is feasible to design an architecture that accommodates the needs of privacy-sensitive users. Using this work as a stepping stone, our future work is to explore a flexible architecture that enables users to select among diverse privacy features.

X. Acknowledgements

We would like to thank the anonymous reviewers for their insightful feedback and suggestions. The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013)/ERC grant agreement 617605; and from the Institute for Information & communications Technology Promotion (IITP) grant funded by the Korean government (MSIP)/No.B0717-16-0040. We also gratefully acknowledge support by ETH Zurich.

References

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

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

[3] E. Fink, "Stalker: A Creepy Look at You, Online," http://goo.gl/DWCiKI, Jun 2014.

[4] M. Wuergler, "Secrets in Your Pocket," https://goo.gl/J16Ykx.

[5] B. Raghavan, T. Kohno, A. C. Snoeren, and W. David, "Enlisting ISPs to Improve Online Privacy: IP Address Mixing by Default," in *P. of PETS Workshop*, 2009.

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

[7] Y. Zhu, X. Fu, B. Graham, R. Bettati, and W. Zhao, "On Flow Correlation Attacks and Countermeasures in Mix Networks," in *P. of PETS Workshop*, 2004.

[8] X. Wang, S. Chen, and S. Jajodia, "Tracking Anonymous Peer-to-Peer VoIP Calls on the Internet," in *P. of ACM CCS Conference*, 2005.

[9] V. Shmatikov and M.-H. Wang, "Timing Analysis in Low-latency Mix Networks: Attacks and Defenses," in *P. of ESORICS Symposium*, 2006.

[10] P. Syverson, G. Tsudik, M. Reed, and C. Landwehr, "Towards an Analysis of Onion Routing Security," in *P. of PETS Workshop*, 2001.

[11] Y. Sun, A. Edmundson, L. Vanbever, O. Li, J. Rexford, M. Chiang, and P. Mittal, "RAPTOR: Routing Attacks on Privacy in Tor," in *P. of USENIX Security Symposium*, 2015.

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

[13] B. Raghavan and A. C. Snoeren, "A System for Authenticated Policy-Compliant Routing," in *P. of ACM SIGCOMM Conference*, 2004.

[14] ARIN, "Resource Public Key Infrastructure," http://bit.ly/1EJCQoT, Jan 2015.

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

[16] J. Viega and D. McGrew, "The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)," RFC 4106, IETF, Jun. 2005.

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

[18] "eBACS: ENCRYPT Benchmarking of Cryptographic Systems," http://bench.cr.yp.to/supercop.html, 2015.

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

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

[21] C. Imbrenda, L. Muscariello, and D. Rossi, "Analyzing Cacheable Traffic in ISP Access Networks for Micro CDN Applications via Content-Centric Networking," in *P. of ACM ICN Conference*, 2014.

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

[23] V. Liu, S. Han, A. Lerner, A. Krishnamurthy, and T. Anderson, "An Internet Architecture Based on the Principle of Least Privilege," University of Washington, Tech. Rep. UW-CSE-12-09-04, 2012.

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

[25] A. Bittau, M. Hamburg, M. Handley, D. Mazières, and D. Boneh, "The Case for Ubiquitous Transport-level Encryption," in *P. of USENIX Security Symposium*, 2010.

[26] M. W. Petullo, X. Zhang, J. A. Solworth, D. J. Bernstein, and T. Lange, "MinimaLT: Minimal-latency Networking Through Better Security," in *P. of ACM CCS Conference*, 2013.