LAP: Lightweight Anonymity and Privacy

Hsu-Chun Hsiao[†] Tiffany Hyun-Jin Kim[†] Adrian Perrig[†] Akira Yamada[‡]
Samuel C. Nelson[§] Marco Gruteser[§] Wei Meng[◇]
[†]CyLab/CMU [‡]KDDI Labs [§]Rutgers University [◇]Tsinghua University

***Abstract*—Popular anonymous communication systems often require sending packets through a sequence of relays on dilated paths for strong anonymity protection. As a result, increased end-to-end latency renders such systems inadequate for the majority of Internet users who seek an intermediate level of anonymity protection while using latency-sensitive applications, such as Web applications. This paper serves to bridge the gap between communication systems that provide strong anonymity protection but with intolerable latency and non-anonymous communication systems by considering a new design space for the setting. More specifically, we explore how to achieve near-optimal latency while achieving an intermediate level of anonymity with a weaker yet practical adversary model (i.e., protecting an end-host's identity and location from servers) such that users can choose between the level of anonymity and usability. We propose *Lightweight Anonymity and Privacy (LAP)*, an efficient network-based solution featuring lightweight path establishment and stateless communication, by concealing an end-host's topological location to enhance anonymity against remote tracking. To show practicality, we demonstrate that LAP can work on top of the current Internet and proposed future Internet architectures.**

I. Introduction

Staying anonymous in today's Internet requires anonymous overlay systems, such as Tor [1], to conceal the communicating endpoint's IP address, as it can reveal the end-user's identity and location [2]. Such overlay systems attempt to facilitate anonymous communication using layer-encrypted packets traveling through indirect routes. However, this results in additional latency due to long end-to-end path length and cryptographic operations indirectly traveling through three Tor relays would be approximately four times slower than traveling along a non-dilated path. Moreover, Tor relays are constantly overloaded [3], further worsening the latency and throughput. Measurements show that the average time to fetch an HTTP header using Tor is 4.04s — ten times higher than fetching it without Tor [4]. Although privacy-anxious users may tolerate seconds of latency for strong privacy, users desiring an intermediate level of privacy for default protection of daily online activities (e.g., prevent websites from tracking them for behavioral advertising[1]) may be impatient to wait.[2]

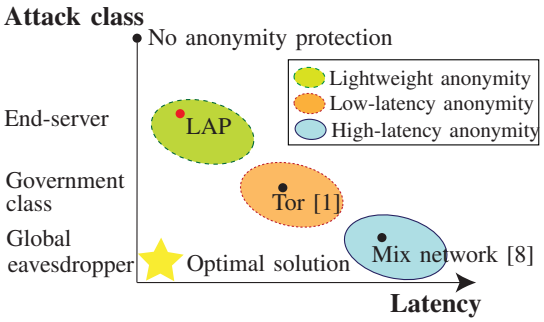


Figure 1. The design space of anonymous schemes.

Despite existing work that attempts to protect end-users' anonymity [1], [7], [8], it still remains a challenge to provide an intermediate level of anonymity and privacy protection without introducing much latency. In this paper, our main goal is to bridge the chasm between systems that provide strong anonymity with high latency and systems that support no anonymity with zero latency, and explore how to support *lightweight anonymity and privacy* that is efficient enough to protect all traffic. Note that those end-users who want an intermediate level of privacy primarily desire to remain anonymous from servers such that servers cannot track their behavior. This implies that guaranteeing the end-user's anonymity and privacy against a single remote entity rather than a strong, global attacker may be a suitable relaxation of the attacker model to gain higher efficiency.

We propose a new setting that we call Lightweight Anonymity and Privacy (LAP-setting for short) for private and anonymous communication in the Internet with the following properties:

- **Low-stretch anonymity:** packets for anonymous and private communication should travel through near-optimal routes such that the increase in the number of Autonomous Domains (ADs) normalized over the original path length is low.
- **Relaxed attacker model:** an intermediate level of privacy can be achieved with sender and receiver anonymity and location privacy. Hence, we relax the strong attacker model (e.g., global or government-class

[1]Users might enable the DO-NOT-TRACK option supported by most mainstream browsers. However, a recent study [5] has shown that this mechanism is hard to use due to configuration complexity and provides no guarantee as it depends on the self-regulation of online organizations.

[2]Studies have shown that online users are sensitive to waiting time: Amazon's sales dropped by 1% for every 100ms increase in page load time, and Google's ad revenue decreased by 20% for a 500ms increase in search result display time [6].

attackers) considered by existing anonymity systems.

As Figure 1 shows, our aim is to address a relaxed attacker model (e.g., end-server attack) with near-optimal latency while existing work addresses stronger attacker models (e.g., government class or global eavesdropper) with higher latency. Although low-latency designs are shown to be inherently vulnerable to a global eavesdropper, some users who trust their local ISPs can achieve much higher efficiency under the LAP-setting.

Our mechanism, *Lightweight Anonymity and Privacy (LAP)*, is an efficient and practical network-based solution featuring lightweight path establishment and efficient communication. LAP attempts to enhance anonymity by obscuring an end-host's topological location, based on two building blocks: packet-carried forwarding state, and forwarding-state encryption.

- **Packet-carried forwarding state:** each packet carries its own forwarding state such that ADs can determine the next hop from the packet without keeping local per-flow state.
- **Forwarding-state encryption:** existing anonymity systems require entire packets to be decrypted/encrypted as they travel using shared keys between the sender and intermediate relays. In contrast, LAP allows each AD to use a secret key (known to the AD only) to encrypt/decrypt forwarding information in packet headers. As a result, an AD's forwarding information can be hidden from all other entities while a LAP packet remains the same at each hop.

LAP is extremely lightweight in the sense that (i) it introduces minimal overhead over non-anonymous packets in terms of latency and computational overhead on routers, (ii) it does not require any per-flow state to be stored on routers, and (iii) no separate keys are required to be set up with routers. In addition to its performance advantages, LAP's unique design provides two additional merits. First, LAP supports different privacy levels such that an end-host can trade privacy for improved performance. Second, LAP is a generic design that can work with a wide range of routing protocols, which includes the inter-domain routing protocol BGP and new proposals such as SCION [9] and MobilityFirst [10]. Furthermore, we show that LAP fits especially well with proposed routing protocols that support packet-carried forwarding state, such as SCION and ICING [11].

In this paper we focus on network-based solutions, where users and locations can be identified through IP addresses. While most current end-host tracking is implemented via cookies [12] and applications may as well leak identifiable information such as email addresses or browser configurations, IP addresses have been used as an alternate identifier when such auxiliary information like cookies is unavailable [13]. Hence, a complete solution for anonymous communication must integrate network-layer techniques with mechanisms for other layers, as recognized by previous network-based proposals [7], [14].

Contributions.

1) We explore the design space of anonymous protocols in the context of a relaxed adversary model.
2) We propose Lightweight Anonymity and Privacy (LAP), an efficient network-based solution that enables lightweight path establishment and efficient forwarding.
3) We evaluate LAP's security and performance advantages. Our systematic analysis and the evaluation of our software implementation confirm that LAP can improve anonymity with low performance overhead.

II. Problem Definition

We study how to camouflage an end-host's topological location (i.e., potential origin within a given topological neighborhood) in a network architecture to enhance anonymity and location privacy in a practical manner. More specifically, we study how to design an anonymous forwarding protocol that can protect the identities and locations of end-hosts from a weaker yet practical adversary, while demanding minimal increase in latency. We do not claim to achieve complete anonymity, but rather focus on providing an intermediate level of anonymity.

In this section, we scope our problem in terms of desired properties, assumptions, and threat model.

A. Desired Privacy Properties

Sender/receiver anonymity. Anonymity can be viewed as being unidentifiable within a set of subjects (e.g., users), also known as an *anonymity set* [15]. This implies that a sender or a receiver can achieve stronger anonymity if its identity is hidden in a larger anonymity set [16].[3] As a result, an attacker cannot link the sender and receiver if either *sender anonymity* or *receiver anonymity* is achieved. Since the design of a full anonymous communication system that can defend against timing attacks and conceal unique platform characteristics is beyond the scope of this paper, we focus on concealing an end-host's network identifier and location in the network topology (which we call "topological anonymity"), which is an important step towards improving sender/receiver anonymity. For simplicity in expression, we also abbreviate "topological anonymity" simply with "anonymity" in the paper.

Session unlinkability. Session unlinkability prevents an attacker from linking a user's activities over time. We want

[3]As Syverson points out, the anonymity set is insufficient to analyze complete sender/receiver anonymity as a thorough analysis with realistic attacker strategies is appropriate [17]. However, we believe that the anonymity set is a tangible metric for evaluating topological anonymity that we aim to achieve in this paper, and we leave it as future work to address various attacker strategies.

to ensure that given two packets from two different sessions, an attacker cannot determine whether these packets are associated with the same sender (or receiver).

Location privacy. Location privacy is achieved when a user conceals her *geographical* location so that an attacker cannot track her whereabouts.

Privacy levels. We want to provide different levels of privacy to end-hosts under end-server attacks in case they are willing to trade privacy for improved performance [18].

In this paper, we consider confidentiality of the packet payload to be orthogonal to the scope of our work as data confidentiality can be achieved using end-to-end encryption. Also, privacy leakage from higher layer protocols/payload is outside the scope of this paper as such an issue can be alleviated by existing tools such as Privoxy.[4]

B. Desired Performance Properties

While providing an intermediate level of anonymity, we want to assure that the anonymity protection introduces marginal overhead. Following are the desired performance properties:

Low path stretch. We define path stretch as the increase in the number of AD hops normalized over the original (or non-anonymity) path length. Since the latency increases as the number of intermediate hops increase on the path, it is desirable to minimize path stretch.

Low performance overhead. We want to minimize cryptographic overhead, especially asymmetric operations and packet decryption and re-encryption at each hop.

Minimal state. To avoid the state explosion problem, we want to keep minimal or no per-flow state to reduce the attack surface and increase scalability.

C. Assumptions

We assume that an end-user trusts her first-hop AD in the sense that the first-hop AD keeps its customers' information private and correctly performs anonymous forwarding protocols. This is aligned with the trust relationship in today's Internet since end-users place more trust on topologically closer ADs and generally have more control over the choice of their first-hop ADs than over the other ADs on a routing path. In case end-users do not trust their first-hop ADs and have no options to pick their own ADs, they may use anonymity systems such as encrypted tunnel IPsec, Tor [1], or anti-censorship systems [19].

We envision that ADs can control the amount of bandwidth allocated for anonymous communication, thus limiting the misuse of anonymous protocols, e.g., for sending untraceable attack traffic. We also assume that routers in ADs support packet-carried forwarding states.

[4]http://www.privoxy.org/

D. Threat Model

An adversary's goal is to break the desired privacy properties described in Section II-A to discover the identity or location of a sender or a receiver of a given packet. More specifically, we focus on topology attacks where an attacker attempts to de-anonymize the sender (or receiver) using topological location information in a given AD-level topology, and leave it as future work to defend against timing correlation attacks [20]–[23].

We consider a relaxed threat model with respect to the attacker's capability: the attacker can compromise any AD except the first-hop AD where the victim end-host resides. Under this model, our primary attack case is an end-server attack where a malicious server analyzes traffic to it or initiate communication with others. We also consider an in-network attack where a malicious AD beyond the first-hop of the victim end-host leverages its cryptographic keys to perform deep packet investigation or actively manipulate (e.g., inject, delete, delay, and replay) packets. Malicious servers and ADs can collude to share their knowledge base.

III. Overview: LAP

LAP is a lightweight protocol to facilitate real-time, bidirectional anonymous communication. In this section we first give a high-level overview of LAP, and explain how end-hosts establish an encrypted path (e-path) and how ADs forward packets along the e-path to achieve an intermediate level of anonymity.

The core observation of this work is that encrypting path information (i.e., concealing forwarding information in the packet header) improves topological anonymity against an adversary in the LAP-setting since the adversary cannot retrieve the sender's (or receiver's) origin address from the packet. Moreover, extending an encrypted path to a benign AD increases the topological anonymity, simply because there are more potential origins whose paths could route through the AD. Extending an e-path beyond one hop is desired because one-hop encryption offers insufficient topological anonymity, as we demonstrate in the preliminary analysis (Section V). We also discuss in Section VII the level of anonymity when the adversary appears at different places on the e-path.

Background: network setting. We consider a network consisting of Autonomous Domains (ADs) as the basic principal of inter-domain routing; each of these ADs has a set of interfaces, each with a unique ID, that can connect to neighboring ADs. ADs agree on an inter-domain routing protocol Θ, e.g., the Border Gateway Protocol (BGP). Upon receiving a packet destined to $dest$, an AD_x evaluates $\Theta_x(dest)$ to determine the next hop of the packet.

Each AD maintains a master secret key, perhaps stored in a secure offline server, and derives short-term secret keys, each associated with a certain time period, from the

master key. The actual encryption and authentication keys are derived from the short-term key and a nonce specified by the sender. We assume every gateway router in an AD has a copy of the short-term keys and knows how to process and route LAP packets within the AD.

LAP overview. At a high level, LAP has two phases, as shown in Figures 2 and 3. Suppose Alice wants to communicate anonymously with Bob without revealing her identity and precise location.

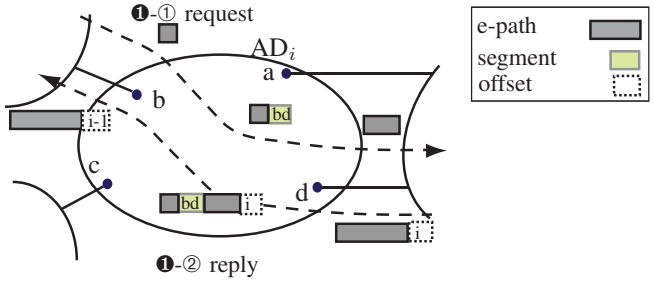


Figure 2. Operations within an AD. Step ❶-①: upon receiving a request packet, an AD encodes its ingress (b) and egress (d) interfaces, extends the e-path in the packet, and forwards the packet (e.g., through interface d in this figure). Step ❶-②: an AD retrieves the interfaces from the e-path in the reply packet and forwards it (e.g., to interface b).

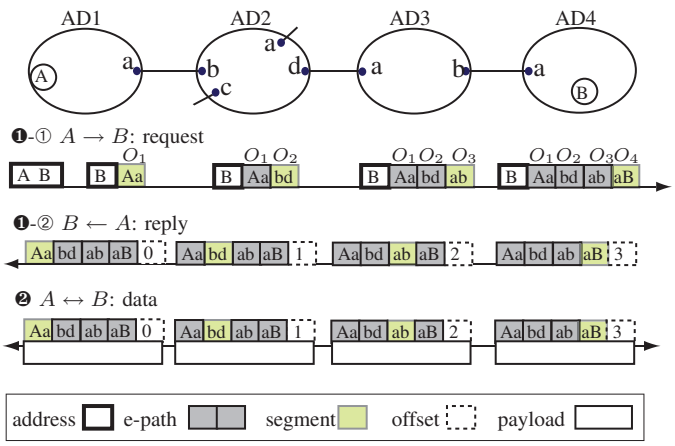


Figure 3. Operations between ADs. Step ❶-①: A sends a `request` to B, which is routed by B's address. Step ❶-②: B replies e-path to A along the reverse path. An AD locates its segment by the offset pointer. Phase ❷: A and B send data to each other along the e-path.

Phase ❶ Establishing e-paths: This phase enables Alice to obtain an e-path — a bi-directional routing path consisting of encrypted forwarding decisions by intermediate ADs on the path.

- **Step ❶-① Request.** To set up an e-path as shown in Figure 3, Alice creates a `request` packet to reach Bob. When her `request` packet reaches a gateway router inside AD_1, it creates a segment which contains Alice's address along with the egress interface, encrypts the segment to anonymize Alice's origin address, and forwards the encrypted segment (O_1) to AD_2. Upon receiving the `request`, as shown in Figure 2, AD_2 encrypts its own forwarding decision in O_2 (i.e., the `request` packet from ingress interface b is forwarded to egress interface d to reach Bob), appends O_2 to the `request` packet, and forwards it to the next AD. This process continues until the `request` reaches AD_4, where Bob resides. Note that encryption and authentication of O_i use secret keys that are only known to AD_i so that only AD_i can later decrypt and verify O_i.
- **Step ❶-② Reply.** The resulting e-path enables Bob to send packets to Alice without knowing her origin address, because the e-path encodes the forwarding decisions made by ADs on the routing path. We leverage *packet-carried forwarding state*, where the network forwards packets solely based on the state contained in the header (i.e., e-path). More specifically, Bob retrieves the e-path from the `request` and puts the e-path in the header of a `reply` packet, which is a special type of `data` packet without payload. As shown in Figures 2 and 3, upon receiving the `reply`, AD_3 decrypts the segment O_3 that it encrypted during Step ❶-①, retrieves the egress interface a, and forwards the `reply` to the next hop. This process continues until the `reply` reaches the intended end-host Alice. If an AD fails to correctly decrypt or verify the segment, the `reply` is dropped.

Phase ❷ Forwarding: When Alice obtains the e-path from the `reply` packet, she can start sending `data` packets anonymously along this e-path using packet-carried forwarding state, as described above.

With LAP, Alice achieves sender topological anonymity and location privacy with respect to a LAP-setting adversary (e.g., Bob), because only her local AD knows her identity and address. In the following sections, we describe LAP in detail, and validate it using a real Internet topology. We also address the challenges of instantiating LAP in the current IP network and future Internet architectures.

IV. LAP: Lightweight Anonymity and Privacy

In this section, we describe in detail how e-paths are constructed, and present additional mechanisms to achieve receiver anonymity and controllable privacy. We start with the packet header formats.

A. LAP Packet Header Format

Figure 4 illustrates the format of a LAP packet header. The header contains a 8-bit Type field to distinguish `request`, `reply`, forward `data` (from Alice to Bob), and backward `data` (from Bob to Alice) packets (six bits of the Type field are reserved for future extensions). The header also contains a 32-bit Nonce field to assist session unlinkability.

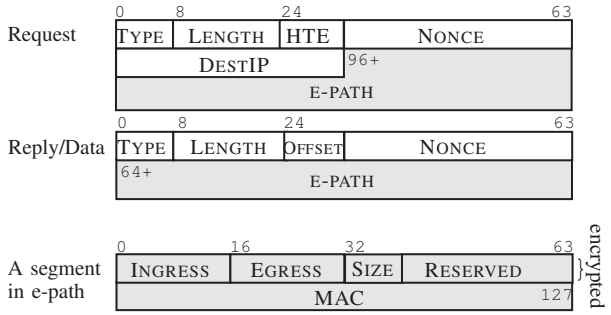


Figure 4. LAP packet header formats. In a segment, the first 64 bits are encrypted, and the RESERVED field can be used to store additional information of an AD.

Request. A `request` packet indicates Alice's intent to anonymously communicate with Bob. To initiate a request, Alice specifies Bob's address in a 32-bit DESTIP field and her desired privacy/performance tradeoff, expressed in a 8-bit HOP-TO-ENCRYPT (HTE) field (to be discussed in Section IV-C). As the request travels through ADs until it reaches Bob, each intermediate AD appends its own encrypted path segment to the E-PATH field (to be described later).

Reply. A `reply`/`data` header contains no IP address since `reply`/`data` packets can be forwarded using the bi-directional e-path that is copied from the corresponding `request` packet. The header also contains a LENGTH field to indicate the size of the packet, and an OFFSET field to indicate the appropriate segment from the E-PATH field that the receiving AD can decrypt. AD_i adjusts the OFFSET field based on the direction of the packet (e.g., for `reply`, OFFSET is decreased by 1).

Segments in e-path. The E-PATH field comprises a sequence of segments, each of which is 128 bits by default. As shown in Figure 4, an AD creates each segment consisting of INGRESS and EGRESS interfaces, size of the segment, RESERVED to store additional information (e.g., source AD can store the source IP address which does not fit in the INGRESS field), and MAC to store the Message Authentication Code over all segments in the E-PATH field (including its own). Note that LAP can support variable-size segments in multiples of 128 bits (and thus a SIZE field is needed in a segment) to defend against size-based passive traffic analysis, as discussed in Section IV-E.

B. LAP Protocol Description

We now describe Phases ❶ and ❷ in detail.

Encrypted path establishment. To construct an e-path, Alice sends a `request` to Bob (Step ❶-①), and by default, LAP requires each AD to append its encrypted routing decision to the received request packet.

Suppose Alice resides in AD_1 and Bob resides in AD_n, and the request packet moves along a path AD_1, AD_2, ..., AD_n. As shown in Figure 3, AD_i generates a segment O_i, which contains the encrypted ingress and egress interfaces for bi-directional forwarding, and appends to the packet. As a result, a resulting e-path $O_{A,B}$ consisting of $\{O_1, \ldots, O_n\}$ is constructed as follows: Let $O_0 = \emptyset$. For $i = 1 \cdots n$,

$$\chi_i = Enc_{k_i^e}(M_i),$$
$$O_i = \chi_i \| MAC_{k_i^s}(\chi_i \| O_{i-1}) \quad (1)$$

where M_i contains an AD's routing decision (i.e., the ingress and egress interfaces), $Enc_k(m)$ means encrypting m using key k, and $MAC_k(m)$ is the Message Authentication Code of m using k. k_i^e and k_i^s are symmetric keys derived from the nonce and the AD_i's current short-term key, known only to AD_i.

We include the previous segment in the MAC computation to enforce the routing decision while preventing attackers from crafting an arbitrary path. Without MACs, an adversary can easily find a ciphertext decrypted to some meaningful egress/ingress interfaces. Simply adding a regular MAC is insufficient because an adversary may be able to craft an invalid path by combining segments obtained from two separate requests. Hence, in LAP, we use layered MACs to prevent arbitrary combinations of segments.

AD_i appends O_i to the E-PATH field of the `request`, and forwards it to AD_{i+1} (via egress interface) until the `request` reaches Bob.

Packet-Carried Forwarding State. For successful packet forwarding using packet-carried state, end-hosts copy the E-PATH field from the preceding packet. For example, upon receiving the `request`, Bob copies the E-PATH field to the `reply` packet. Similarly, when Alice receives the `reply`, she copies the E-PATH to the `data` packet, and Alice and Bob copy the E-PATH for succeeding `data` packets.

Using the e-path in a `reply` packet (Step ❶-②) and a `data` packet (Phase ❷), ADs can forward the `reply`/`data` packet along the encrypted path bi-directionally without actually knowing Alice's or Bob's address. Suppose a `reply` packet enters an AD_i from interface d, as shown in Figure 2. The AD proceeds as follows:

1) *Retrieve forwarding decision:* It first locates its segment O_i based on OFFSET and TYPE (which encodes the direction of forwarding) in the header. It then decrypts χ_i to recover the ingress interface ig, egress interface eg.
2) *Verification:* O_i is valid if the following conditions hold: i) For a forward packet (e.g., `data`), $d = ig$; for a backward packet (e.g., `reply`, `data`), $d = eg$, ii) MAC verification succeeds (i.e., the AD re-computes the MAC using its current secret key and the information embedded in the header, and checks if the resulting MAC matches the one included in O_i.)
3) *Forwarding:* If this segment is valid, the AD determines the exiting interface and adjusts the offset.

In our example, since this is a backward packet, the exiting interface is ig and the offset should be decreased by 1. The AD then forwards the packet to the exiting interface.

Since ADs rotate their short-term keys periodically (e.g., every hour) for security, Alice may have to renew or request a new e-path if any key for decrypting or verifying the e-path expires during her session. LAP can support efficient renewal by embedding updated e-path in data packets.

Session unlinkability. Alice can request a new e-path (by specifying a different nonce) for every new session to achieve session unlinkability. Also, the encryption algorithm should be secure against chosen-plaintext attacks such that encrypting the same plaintext twice would result in two different ciphertexts with high probability. For example, one can use AES in CTR mode. The initialization vector (IV) in CTR mode can be derived from the nonce and the previous O_i to avoid allocating extra space for storing IV in the packet. Since a different nonce or routing path would result in a new e-path, an attacker has a low success rate in correlating two separate sessions based on an e-path.

C. Controllable Privacy Levels

Encrypting every AD hop in LAP increases the packet header size and computational overhead, and may reduce the flexibility in routing (e.g., in the case of multipaths, the sender cannot make an informed decision in path selection without knowing which ADs are on the path.) Although LAP provides negligible computational overhead on routers (see Section VIII) and we anticipate that routers will be improved to support larger packets, users may still want to trade privacy for improved performance.

LAP provides options for end-hosts to control the length of e-paths, which results in differentiated privacy disclosure. The intuition is that the degree of anonymity and privacy (in terms of the size of an anonymity set) increases with the length of an e-path (in terms of the number of AD-hops). More specifically, Alice specifies the desired length of the e-path in a Hop-to-Encrypt (HTE) field in the `request` packet. Each AD checks the HTE field before updating the e-path, and if HTE ≥ 1, the AD updates the `request` packet as usual and decreases the HTE field by 1. If HTE reaches zero before reaching Bob, the intermediate AD returns the e-path to Alice on a `reply` packet. Similarly, if Bob receives the packet with HTE ≥ 0, Bob returns the e-path to Alice on a `reply` packet. Note that to use such partially encrypted paths, packets have to contain an extra field storing the destination's address (which, however, can be in plaintext, as receiver anonymity is provided using rendezvous points, as will be explained in Section IV-D). During the forwarding phase, the AD at the end of the e-path converts `data` packets between the LAP- and regular-mode. For example, in BGP routing, the AD encapsulates the e-path in a normal IP packet and sets the source address to be its own address and the destination address to Bob's.

D. Path Publishing for Receiver Anonymity

We have shown that Alice can achieve sender anonymity and location privacy by constructing an e-path to Bob (i.e., only Alice's first-hop AD knows her identity and location). However, sometimes Bob may want privacy protection as well. For example, a user running a controversial website (e.g., WikiLeaks) would prefer to hide his location and permanent identity to prevent tracking or avoid censorship. However, since a receiver is unaware of who a sender might be in advance, the challenges become (1) how the receiver constructs an e-path for any potential sender and (2) how a sender looks up the receiver's e-path without knowing his permanent identity.

At a high level, to achieve receiver anonymity, Alice and Bob each initiate an e-path to a *rendezvous point* so that only the local ADs know the identity of end-hosts. Such an indirection technique is commonly used in anonymity systems [1]. To address the second challenge, Bob publishes his e-path associated with his pseudonym on a publicly-accessible *path server*. As a result, a sender knowing Bob's pseudonym (e.g., via out-of-band communication) can retrieve Bob's e-path from the path server and reach Bob through the rendezvous AD. In theory, any AD in the Internet could be a rendezvous point or host a path server. To minimize the path stretch and communication overhead, in practice, tier-1 ADs are a reasonable choice of rendezvous ADs and path server administrators, because most of the Internet traffic goes through tier-1 ADs.

E. Padding Against Size-Based Traffic Analysis

If we use fixed-size segments, an attacker can determine the distance (in terms of AD hops) to a sender based on the size of the header. Hence, LAP allows ADs to pad segments (variable-size segments) to enhance topological anonymity. As mentioned in Section IV-A, the size of each variable-size segment is in multiples of 128 bits. For proper decryption and adjustment of the offset, each AD needs to know the size of its own segment. Hence, to allow proper operations on both forward and backward packets, an AD using a variable-size segment encodes the size in both the first and last 128-bit blocks in the SIZE field as follows: AD_i (1) creates the first 128-bit block O_i using symmetric key k_i as described in Section IV-B; and (2) copies the same INGRESS, EGRESS, and SIZE to the last 128-bit block of its segment, and creates the MAC over the entire segment using another symmetric key k_i'. In this manner, the first 128-bit block looks different from the last 128-bit block. With this process, the AD can recover the length of its own segment from either the first or the last 128 bits of the segment, and adjust the offset properly. For (1), note that since an AD does not know the

size of the previous segment, it computes a MAC over the last 128 bits of the previous segment.

With these variable-size segments, an attacker can only obtain an upper bound on the distance to the sender, which is the size of the e-path in bits divided by 128. The optimal way of padding results in an e-path of $128 \cdot l$ bits, where l is the distance of the farthest potential sender in AD hops.

V. Preliminary Analysis

In this section, we illustrate that the current Internet provides minimal anonymity, and demonstrate how LAP can increase the level of anonymity with a real Internet topology.

A. Anonymity and Privacy in the Current Internet

Anonymity in the Internet is hindered by long lasting end-host identifiers, namely IP addresses. From a network layer's perspective, IP addresses identify both the source and the destination of the traffic. Hence, by snooping on traffic flows, malicious nodes can easily determine which end-hosts are communicating with each other and link different sessions to the same end-hosts. While public servers prefer long-lasting IP addresses for availability, current Internet protocols and ISP policies generally assign IP addresses that last on the order of days [24] to clients who have no desire to run public servers. Typically, these IP addresses (from the ISPs allocated address space) change only when the DHCP lease time expires. While NAT boxes can provide an anonymity set greater than one, devices behind them are usually both small in number and in the same geographical area, thus providing extremely limited privacy guarantees. In the cellular realm, the situation is better since providers' NATs can mask a wider range of clients [25]. Ideally, privacy solutions should be available in all domains that easily allow end-hosts to retain anonymity at the network level.

Consequently, while the current Internet intrinsically provides a certain level of anonymity based on dynamic addressing techniques (e.g., DHCP and NAT), the degree of anonymity is constrained by the size of the IP prefixes. More specifically, we estimate the anonymity set size by analyzing the announced prefix sizes and the number of subscribers of six main ISPs in the U.S., as Table I summarizes. We group the prefixes (extracted from the RouteViews dataset [26]) into ISPs using AS description from the CIDR report.[5] Assuming that subscribers are uniformly distributed in an ISP's address space, the size of an anonymity set can be as low as $2^{4.7} \simeq 26$.

Similar studies have shown that hiding behind a prefix provides insufficient anonymity [7]. Although aggregating prefixes associated with the same location may increase the size of the anonymity set (but not location privacy), the flexibility of route management within an ISP may diminish. Also, users have no control over their level of anonymity.

[5]http://www.cidr-report.org/as2.0/

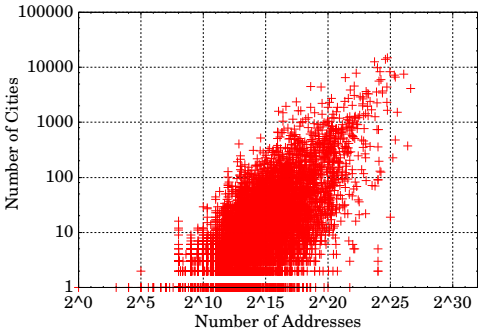


Figure 5. As the number of possible IP addresses increases, so does the number of potential cities.

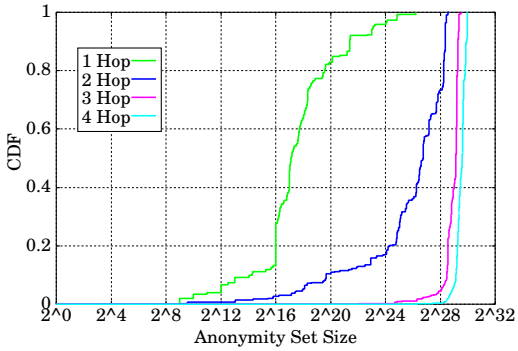


Figure 6. Comparison of anonymity set size based on the number of encrypted hops. As the number of encrypted AD hops increases, the anonymity set size increases. For the case of 4 encrypted hops, almost all origins enjoy an anonymity set size of over 2^{28} hosts.

We also investigate location privacy in the current Internet. We use the Maxmind GeoIP locationing tool to estimate an end-host's current city[6] based on its IP address and quantify the location-privacy level based on the number of cities the end-host may reside in. Figure 5 shows the relationship between the number of cities and the anonymity set size: the level of location privacy can be increased by increasing the number of possible IP addresses.

B. Anonymity in LAP

In LAP, users can improve their anonymity set size by extending the length of their e-paths.

To show the effectiveness of LAP path encryption, we evaluate anonymity in LAP using traceroute data from iPlane's measurements and routing data from RouteViews [26]. The iPlane dataset contains traceroute data between 197 sources and about 13 thousand destinations. We eliminate 28 sources with incomplete logs and choose 1,000 destinations for each source. For each pair of source and destination, we calculate the size of the source anonymity set with respect to the destination based on the Internet topology and the assigned address space extracted from the

[6]Maxmind determines city names based on the Geographic Names Data Base. http://www.maxmind.com/

Table I
ANONYMITY SET SIZE OF US TOP ISPS.

ISP	Address Space (Entropy)	Announcing Prefix	Subscriber [27] (Entropy)	Subscriber Entropy/Prefix		
				Ave.	Min	Max
Comcast	70,374,912 (26.1)	865	17,406,000 (24.1)	19.5	6.0	22.0
Time Warner	27,556,352 (24.7)	2,158	9,992,000 (23.3)	14.4	6.5	17.5
Cox	11,971,584 (23.5)	1,507	4,400,000 (22.1)	18.8	6.6	19.6
ATT	114,544,128 (26.8)	6,127	16,485,000 (24.0)	18.3	5.2	21.2
Verizon	84,403,200 (26.3)	4,376	8,490,000 (23.0)	15.5	4.7	19.7
Quest	84,403,200 (24.0)	899	2,965,000 (21.5)	16.2	5.5	18.5

RouteViews dataset. According to Figure 6, which illustrates the CDF (cumulative distribution function) of the number of addresses, the increase in the number of encrypted hops increases the anonymity set.

VI. LAP INSTANTIATION

In this section, we discuss how LAP can be accommodated in the current IP network running BGP. We then discuss the potential benefits of tailoring LAP to two future Internet architectures: SCION [9] and MobilityFirst [10].

A. LAP in the Current Internet

In this section, we delineate how LAP can be incrementally deployed in the current IP network. We consider both LAP-enabled ADs and legacy ADs that do not support LAP. In such heterogeneous networks, one main challenge is to enable a LAP-enabled AD to discover and build virtual channels to nearby LAP-enabled ADs. For this integration, we assume that the IP header contains a *LAP-flag* bit that is set if an IP packet encapsulates a LAP packet.[7]

A legacy AD is agnostic to the encapsulated LAP packet and routes IP packets based on the destination IP as specified in the IP packet header. A LAP-enabled AD, on the other hand, installs dedicated LAP routers where each of them has a publicly-accessible address, and configures every gateway router to route LAP packets (whose LAP-flag is set) to the nearest LAP router. Figure 7 illustrates a scenario where AD_1 and AD_3 are legacy ADs, and AD_2 and AD_4 are LAP-enabled ADs. X and Y represent the LAP routers in AD_2 and AD_4, respectively.

When Alice (whose IP address is A) wants to diffuse her topological location for her communication with Bob (whose IP address is B), she installs a LAP application proxy on her machine. To obtain an e-path, this proxy prepares a LAP `request` packet and encapsulates it in an IP packet. Then, this IP packet is initiated with $srcIP = A$ and $destIP = B$.

[7]Several potential approaches exist to add LAP to the current IP header. One approach would be to add a LAP IP options field, however, this would constrain the length of the LAP header and possibly also slow down packet processing at legacy routers. Another approach would be to use a bit in the current IP header to indicate presence of a LAP header. We could use bit 0 of the 3-bit FLAGS field, which is currently unused. Another potential use could be a bit within the TYPE OF SERVICE OR DIFFERENTIATED SERVICE byte, since the PRECEDENCE or the ECN bits are rarely used. Yet another approach would be to set the PROTOCOL field to indicate that the next header is a LAP protocol header. In the two latter cases, the LAP header could be placed between the IP and TCP or UDP headers.

Encrypted path establishment. The `request` packet sets up an anonymous return path by which Bob can reach Alice without knowing her IP address. When a gateway in the LAP-enabled AD_2 receives a LAP-flagged `request` packet, it routes the packet to the dedicated LAP router X. X then encrypts the $srcIP$ to generate its e-path segment O_2 and appends O_2 to the encapsulated LAP packet. X also updates the $srcIP = X$ in the IP header but $destIP$ remains the same. Similarly, AD_4 process the packet in the same way. When Bob, receives a packet whose $srcIP = Y$ and $destIP = B$, he sends a `reply` packet with $srcIP = B$ and $destIP = Y$. We assume that the LAP-flag and LAP header are preserved in the `reply` packet. When router Y receives the `reply`, it verifies O_4, extracts the IP of the previous LAP router (i.e., X) from O_4, and updates the destination address to be X. Similarly, router X retrieves A from O_2 and updates $destIP = A$.

Forwarding. Alice obtains an e-path from the `reply` packet. To send a data packet to Bob, Alice prepares a LAP `data` packet that contains the e-path and encapsulates it in an IP packet whose $srcIP = \emptyset$ and $destIP = B$. Upon receiving a LAP data packet, Bob returns data packets using the embedded e-path, as described above. Note that ADs can distinguish forward and return `data` packets based on the TYPE field and adjust the OFFSET correctly.

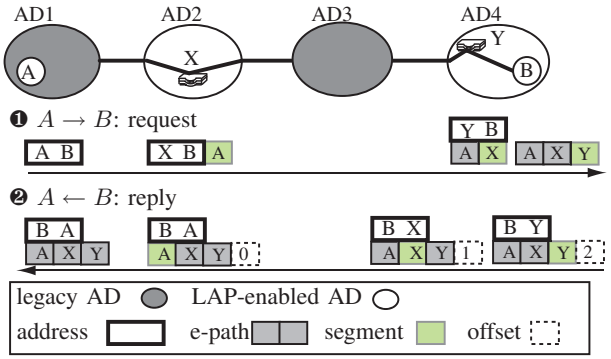


Figure 7. Incremental deployment.

Asymmetric paths. Another advantage of LAP integrated with the current Internet is that it can support asymmetric inter-domain paths, which may exist in BGP due to routing policies, because in this instantiation LAP path is defined by a list of IP addresses instead of interfaces.

B. Integrating LAP into SCION

In this section, we show that LAP can be seamlessly integrated into SCION [9], a high-availability network architecture. LAP only requires an overhead for path establishment and encryption/decryption of packet-carried forwarding information, because packet-carried forwarding state and encrypted path publishing/downloading can be embedded into the existing SCION framework.

Background of SCION routing. SCION groups ADs into Trust Domains (TDs), where each TD aggregates ADs that agree on a common root of trust, usually mapping to an area of uniform legal jurisdiction. Each TD has a TD core consisting of the tier-1 ISPs within this trust domain. TD cores are in charge of two tasks. First, they periodically broadcast Path Construction Beacons (PCBs) by which an AD can learn one or more paths to/from this AD's TD core. Second, TD cores manage authoritative servers such as SCION path servers. Upon receiving PCBs, an AD selects multiple paths along which it can be reached from its TD core, and publishes these (downstream) paths to a path server. To create an end-to-end routing path, a source queries a path server for the destination's downstream paths and then splices one of its upstream paths with one of the destination's downstream paths.

Path encryption requests. SCION ADs route packets using (unencrypted) packet-carried forwarding state and verify the forwarding information using MACs. Hence, running LAP with SCION requires adding symmetric encryption/decryption functions to routers. In SCION, a source obtains a set of paths to reach the destination for source-selection routing. Hence, Alice embeds a `request` packet inside a SCION packet by specifying one of the (unencrypted) paths for an e-path construction. Upon receiving this packet, an intermediate AD (AD_i) appends its O_i and removes the (unencrypted) previous hop information to erase the trace. The `reply` and `data` forwarding can be done as described in the LAP protocol section (Section IV-B).

Path server and rendezvous points in TD cores. The design of SCION requires a path server to store ADs' downstream paths, as an end-to-end path is constructed by splicing a source-to-core path with a core-to-destination (downstream) path. Similarly, LAP also requires a path server that stores encrypted paths to certain rendezvous points. Hence, SCION path servers can manage both SCION paths and LAP's encrypted paths. In this manner, a TD core becomes a default rendezvous point since all paths can traverse the TD core. Note that for the sake of efficiency (shorter paths), SCION may permit shortcuts that bypass a TD core by comparing and finding the intersection of the upstream and downstream paths. However, in LAP, finding such common intersections (common links or ADs) when the intersections are encrypted is fundamentally infeasible because an attacker could take the intersection finding algorithm as an oracle to decipher encrypted paths. Fortunately, the semi-encrypted paths (constructed by setting a small HOP-TO-ENCRYPT value in the `request`) in LAP enable part of a path to be encrypted for a sufficient degree of privacy with the other half remaining unencrypted to enable shortcut construction.

C. Integrating LAP into MobilityFirst

To further illustrate the flexibility of LAP, we now describe how it can also be integrated into a mobility-centric future Internet architecture called *MobilityFirst* [10]. MobilityFirst retains a distributed routing control plane similar to that of BGP, while providing a clean separation of network "entities" and routable addresses. Privacy is a major concern for mobility-centric architectures since they allow humans, via devices they carry or drive, to be continuously connected to the broader Internet. Hence both control-plane reachability updates as well as content generated by these devices have the potential to breach privacy. Low-stretch privacy solutions that cleanly integrate with mobility-centric architectures can give end users privacy with minimal disruption. As with SCION, LAP naturally complements MobilityFirst and adds little overhead.

Background of MobilityFirst. MobilityFirst is a clean-slate Internet architecture designed to address challenges brought about by an increase in the number of mobile, wireless devices. At its core, MobilityFirst provides a mechanism to abstract network entities important to applications, and bind those abstractions into routable network addresses. Specifically, entities such as an individual laptop, a vehicle, a piece of content, or a group of people each obtain a *globally unique identifier*, or GUID, that the application uses for communication. When data destined for a GUID is received by a MobilityFirst router, the router will either attempt to directly route on the GUID or bind the GUID to a routable address via a massively distributed *global name resolution service*, or GNRS. All publicly available entities are responsible for ensuring that their GUID-to-network address mapping is up-to-date in the GNRS. The GNRS is accessible from all MobilityFirst routers and hence GUIDs can easily be re-bound deeper in the network if the destination's network address has changed. In addition to separating naming from addressing, MobilityFirst heavily utilizes in-network storage and hop-by-hop transfer of large data chunks to react to network and host mobility.

Path encryption requests. MobilityFirst's low-level routing plane is similar to that of BGP, with the exception of IP prefix announcements. Since the GNRS handles the "who is in what network" question, MobilityFirst routing simply needs to exchange AD-level reachability information. A LAP path encryption request will occur after a MobilityFirst router (e.g., the border router of the source AD) queries the

destination GUID for a destination network address. The destination network address can be used as the destination of a path encryption request. This process, as described in Section IV-B, can then proceed as it would with BGP.

Path server and rendezvous points. The GNRS is responsible for binding GUIDs to routable addresses, and hence is a perfect match for the LAP path server. Using LAP, the GNRS will bind a GUID (which may be a pseudonym) to an e-path leading to a rendezvous point. Therefore, a router wishing to route towards a destination GUID will make a GNRS query and either get back the destination network address or an e-path leading to a rendezvous point. MobilityFirst networks, however, do not have a strict hierarchy, and hence choosing a rendezvous point is less intuitive. However, since the GNRS is capable of handling multi-homed GUIDs, multiple rendezvous points can be uploaded and bound to the same GUID. If the destination also provides hints, such as "use encrypted path 3 if in North America", this can alleviate stretch problems at the expense of some decrease in location privacy.

Handling mobility. In order to dynamically respond to mobility and disconnection deep within the network, the destination GUID is always available as the authoritative header on a piece of data. Routers detecting a problem with a destination network address can always query the GNRS and re-bind the GUID to a new destination address. LAP integration does not change this, as the destination GUID can always be re-bound to a new e-path obtained from the GNRS.

VII. Security Analysis

We analyze how LAP conceals end-hosts' topological locations for an intermediate level of anonymity, and achieves session unlinkability. We also describe how LAP defends against attacks.

A. Sender/Receiver Anonymity Analysis

In this analysis, we consider a scenario where Alice and Bob communicate with each other along an AD path AD_1, AD_2, $\cdots$, AD_n and quantitatively analyze the degree of anonymity with respect to an adversary, adv, at various vantage points on the path.

We compare LAP with three related anonymous systems: Tor [1], Tor Instead of IP [14], and AHP [7]. We show that LAP provides a competitive degree of anonymity compared to low-latency anonymity systems in the presence of LAP-setting adversaries. Also, LAP guarantees much stronger anonymity properties compared to AHP, which provides a limited level of protection due to a small anonymity set and does not support receiver anonymity.

Notation. We denote $\mathbb{A}_s^{adv}(x)$ as the sender anonymity set of user x with respect to adversary adv. The receiver anonymity set $\mathbb{A}_r^{adv}$ is defined similarly. Let N be the total number of Internet users; thus, N is the maximum size of an anonymity set. N_t is the number of Tor users and $N_t \leq N$. In practice, $N_t \ll N$ because N_t is between 10^5 – 10^6 [28] while N is on the order of 10^9 [29].

Assumptions. As mentioned in Section II-A, a sender can achieve stronger anonymity if its identity is hidden in a larger anonymity set. For the analysis, we assume equiprobability for subjects in an anonymity set. That is, an adversary can determine who may have sent or received a packet within a given anonymity set but cannot tell whether one is more likely to send/receive than the others in the same set. We consider a LAP-setting adversary, who can leverage topological information but not timing information and cannot compromise the first-hop AD of a victim. An adversary with the knowledge of the AD-level topology can narrow down the anonymity set of a packet based, for example, on the length of the packet header and the packet's incoming interface. For this analysis, we assume full deployment of LAP, Tor Instead of IP, and AHP.

We summarize our analysis in Table II, where the first column describes the adversary's location and the following columns present ($|\mathbb{A}_s^{adv}(Alice)|$, $|\mathbb{A}_r^{adv}(Bob)|$) for LAP, Tor, Tor instead of IP, and AHP. Below, we justify the table.

1) LAP: In this analysis, we consider LAP with full path encryption (Alice's e-path + Bob's e-path through a rendezvous AD AD_v in Tier 1) and optimal padding. Hence, a malicious AD can conclude that the sender (or receiver) must reside in an AD that is reachable from the incoming (or outgoing) interface. However, because of optimal padding, an attacker cannot obtain identifiable information from the size of the header.

In LAP, only the first- or last-hop AD knows the identity of the sender or receiver, respectively. Hence an adversary cannot link the sender and the receiver in LAP unless he controls both the first and the last ADs along the path (adv8 in Table II), which is, however, outside our threat model. Moreover, the degree of anonymity increases with the length of the e-path. In other words, the farther away an attacker is from the user, the higher the degree of anonymity. For example, if Bob is an attacker (adv1 in Table II), Alice's sender anonymity set is N, because Bob has no knowledge of the interface information, and every Internet user could be the sender from Bob's point of view. On the other hand, if Alice's first-hop AD is the attacker (adv7), her anonymity set is 1.

Generally, the degree of anonymity strictly increases as the attacker's position moves toward AD_v (adv3), because for each additional AD between Alice and the attacker, users in that AD are added to the anonymity set:

$$|\mathbb{A}_s^{AD_i}(A)| \geq |\mathbb{A}_s^{AD_j}(A)| + |AD_j|$$
$$\Rightarrow |\mathbb{A}_s^{AD_i}(A)| > |\mathbb{A}_s^{AD_j}(A)| \text{ if } v+1 \geq i > j$$

Table II
COMPARISON OF SENDER AND RECEIVER ANONYMITY, REPRESENTED BY THE PAIR OF ($|\mathbb{A}_s^{adv}(Alice)|$, $|\mathbb{A}_r^{adv}(Bob)|$). ASSUME FULL DEPLOYMENT OF LAP, TOR INSTEAD OF IP, AND AHP. $|AD_x|$ IS THE NUMBER OF CLIENTS IN AD_x.

	Adversary adv		LAP	Tor [1]	Tor instead of IP [14]	AHP [7]
LAP-setting	adv1	AD_n	(N, n/a)	(N_t, n/a)	(N, n/a)	($\leq |AD_1|$, n/a)
	adv2	AD_i ($v < i < n$)	(N, $< N$)	(N_t, N_t)	(N, $< N$)	($\leq |AD_1|, 1$)
	adv3	AD_v (or Tier 1)	($\approx N, \approx N$)	(N_t, N_t)	(N, N)	($\leq |AD_1|, 1$)
	adv4	AD_i ($1 < i < v$)	($< N, N$)	(N_t, N_t)	($< N, N$)	($\leq |AD_1|, 1$)
	adv5	AD_1	(n/a , N)	(n/a, N_t)	(n/a, N)	(n/a, 1)
non-LAP-setting	adv6	AD_n	(N, 1)	(N_t, 1)	(N, 1)	($\leq |AD_1|, 1$)
	adv7	AD_1	(1, N)	(1, N_t)	(1, N)	(1, 1)
	adv8	adv6+adv7	(1,1)	(1, 1)	(1,1)	(1, 1)

If the attacker is beyond AD_v (adv4), the anonymity set is $|\mathbb{A}_s^{AD_{v+1}}(A)|$ because the rendezvous AD is known. That is, $|\mathbb{A}_s^{AD_i}(A)| = |\mathbb{A}_s^{AD_{v+1}}(A)|$ if $i > v+1$. Therefore, when the attacker is on the path between Bob and AD_v (including Bob), Alice has the highest degree of anonymity, where any end-host in the network could be the sender (assuming that AD_v is reachable from all end-hosts).

Finally, colluding ADs can easily share knowledge and correlate packets since LAP does not conceal packet content and packet size. Thus, the resulting anonymity set is the intersection of those perceived by individual malicious ADs. Also, LAP provides no anonymity if both end-point ADs collude.

2) Tor [1]: For the purpose of this analysis, we assume that Alice and Bob are Tor clients but do not serve as Tor relays. An attacker can learn a list of Tor relays from Tor directory servers. Hence Alice's first-hop AD (AD_1) can observe that she is sending packets. However, the second-hop AD (AD_2) cannot learn the origin of the packet because it cannot distinguish whether the Tor sender resides in AD_1, or the packet is relayed by other Tor servers and routed through AD_1. In general, if an attacker is an AD except AD_1, Alice is hidden within all active Tor users (N_t). The same analysis can be applied for receiver anonymity. Unlike LAP, Tor can prevent colluding ADs from linking Alice with Bob based on topological or packet information, because layered-encrypted packets look different at each AD. However, Tor is vulnerable to timing attacks performed by colluding ADs (e.g., adv 8).

3) Tor Instead of IP [14]: Recent proposals identify the importance of improving the default privacy level at the network layer. Instead of using Tor as an overlay, Liu et al. propose replacing IP with Tor. They assume that each AD runs a Tor server, and that packets travel from the sender to the Internet core (Tier 1) and then to the receiver similar to LAP rather than being routed via an indirect path. Tor instead of IP, however, allows zigzag paths in the core to improve anonymity. Hence, this scheme exhibits the same level of anonymity as LAP when an attacker is not at the core, but a slightly better anonymity when the core AD is malicious. However, in terms of performance, this scheme suffers from expensive path establishment and stateful communication similar to Tor.

4) AHP [7]: Raghavan et al. propose Address Hiding Protocol (AHP), in which an ISP shuffles its own address space and assigns a random IP to a sender. Trostle et al. present a similar approach to enhance sender's location privacy using Cryptographically Protected Prefixes (CPP) [30]. Both AHP and CPP achieve a level of sender privacy constrained by the available address block and geographical distribution of the sender's hosting ISP. For example, the sender anonymity in AHP is bound by the size of the first-hop AD (or ISP). Also, they do not offer receiver anonymity or location privacy.

B. Session Unlinkability

Session unlinkability can be achieved by requesting a new e-path for every new session. Furthermore, a sender can refresh paths more frequently or use more than one path simultaneously, thanks to the lightweight construction of an e-path. Hence, LAP does not require the same path to be reused for multiple TCP sessions. We show that LAP achieves session unlinkability by considering the knowledge of a malicious AD in the LAP-setting as follows. From a `request` packet, an AD knows an e-path to the sender, the size of e-path (which provides an upper bound on the AD-level distance to the sender), the receiver's ID (say, Bob), and its own segment. A malicious AD can store this information in his own local database. Upon receiving a `reply` or `data` packet, the malicious AD can compare the stored segments from the e-path in the packet, and learn the missing segments from the sender to the receiver. As a result, all `data` packets carrying the same segments would be linked to the same sender-receiver session. On the other hand, when different segments are used in a new session, the AD cannot tell if Bob is still communicating with the same sender, thus achieving session unlinkability.

C. General Attack Resilience

DoS resilience. Prior anonymity systems are often vulnerable to computational-based DoS due to expensive asymmetric operations for setting up communication paths or storage-based DoS due to stateful forwarding. As a result, they require additional DoS defense mechanisms, such as introduction points [1] or mailboxes [14], as an extra layer of indirection to actively block unwanted requests. On the other

hand, LAP is robust against Denial-of-Service (DoS) attacks in many aspects, thanks to its lightweight path establishment and stateless forwarding mechanism. For example, a receiver can filter incoming traffic by selectively announcing paths and frequently updating paths.

A common challenge for all anonymity systems is when an attacker sends untraceable traffic. To prevent such misuse of anonymous communications, an AD can allocate only a small amount of bandwidth for anonymous traffic. To prevent such attacks, we leave it as future work to study the tradeoffs between anonymity and accountability.

Resilience against traffic analysis. Traffic analysis comprises two parts: observing traffic and correlating traffic. Compared to Tor, LAP makes correlations much easier but observations much harder. For correlations, an attacker controlling two or more distinct entities in the network can easily correlate observed packets to estimate their routes, because LAP packets in the same session look the same at each hop. For observations, a Tor attacker controlling all entry and exit relays has a good chance of de-anonymizing Tor traffic. However, the equivalent attack is almost impossible in LAP because the attacker has to compromise all the first-hop ADs.

D. Resilience against Known Attacks

DoS-based side-channel attacks. In the category of DoS-based side-channel attacks, the approach proposed by Burch and Cheswick [31] for IP traceback could also be applied to trace back an e-path to its origin. The basic idea is to send a large amount of traffic over a link that the e-path may be using. If the link is indeed part of the e-path, one will observe a slowdown of the session using the e-path. By repeating this process, one could eventually trace back the entire path. The essence of the approach is to induce a DoS attack and to use other packets as a side channel to determine the packet flow. Numerous such side channels have been investigated in the literature [20], [21], [23], [32]. Flow watermarking techniques also fall into this attack category, using slight time-based variations to infer which packets belong to the same session [33], [34] — however, this requires multiple observation points in the network. These attacks are possible even on more heavy-weight schemes such as Tor, and naturally our lightweight approach will not offer protection. These attacks, however, require more significant effort than passive observations of network traffic.

Time-based identity inference attacks. A related attack class is time-based identity inference attacks. Specifically, Kohno et al. propose device fingerprinting based on clock skew inferred from TCP timestamps [35]. Since in LAP, TCP headers are not encrypted by default, this attack would apply; however, the standard countermeasures apply as well: end-to-end IPsec tunnel, perturbation of TCP timestamp, etc. Another potential location leak is round-trip-time (RTT) based location inference, where the observation is that the lowest observed RTT induces an upper bound on the distance of the other party. Consequently, ACK packets for example, may need to be delayed to increase the anonymity set.

TTL-based attacks. Finally, in the case of LAP used on IP-based networks, we need to defend against a TTL-based attack: by sending a LAP packet with a small TTL, the TTL may expire while a router within the e-path forwards the packet, which in turn would trigger an ICMP message sent to the source address. Fortunately, the first router in the e-path sets the IP source address to its own address, thus the attacker would not receive the ICMP error message.

VIII. Evaluation

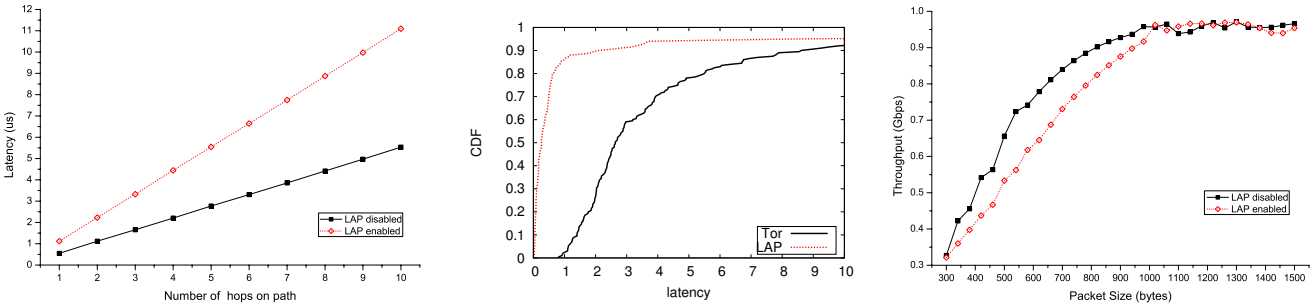
In this section, we evaluate the performance of LAP in terms of latency and throughput. Specifically we compare three systems: LAP-disabled (no anonymity), LAP-enabled (intermediate anonymity), and Tor (high anonymity). Our results show that LAP improves anonymity with a negligible overhead (i.e., lightweight) and is more efficient compared to high anonymity systems like Tor.

LAP implementation. We implement basic routing and forwarding elements based on Click software routers [36] to support packet-carried forwarding state (LAP-disabled). We extend the prototype to further support encryption/decryption of LAP (LAP-enabled). The only overhead that LAP introduces for an e-path construction per AD hop is the extra packet space needed for optimal padding, and the time for a symmetric encryption. This is because packet-carried forwarding state already requires ADs to verify their own routing decisions using MACs. Since routing decisions are carried in each packet, the overhead caused by the forwarding phase for each AD is the time to decrypt its own segment. We show that our software-based implementation of LAP exhibits competitive performance, with an anticipation that LAP will perform even better on dedicated hardware.

A. Latency Evaluation

We first examine the latency introduced by LAP's cryptographic operations. We then estimate LAP's latency in the real Internet and compare with Tor.

We measure the latency of LAP-disabled and LAP-enabled systems in one LAN network. Each AD is simulated on one machine with 1 Gbps connection to its adjacent ADs. Since our tests are run on a local LAN, the latency is dominated by the cryptographic operations. We implement LAP's encryption/decryption using the AES function in OpenSSL. For the LAP-disabled case, ADs perform forwarding using packet-carried state, which involves one MAC computation using the same AES function. For the LAP-enabled case, ADs verify a MAC and decrypt their own state during



(a) Average latency with LAP disabled and LAP enabled.

(b) Latency comparison of LAP and Tor using the real Internet topology.

(c) Average throughput with LAP disabled and LAP enabled.

Figure 8. LAP evaluation.

forwarding. We run each test 10 times and present the average value. As Figure 8(a) shows, LAP adds a small amount of latency to packet processing; in our software implementation, this is on the order of microseconds, but a hardware implementation would shrink the extra decryption time to nanoseconds.

We also compare the latency experienced by LAP and Tor users using the real Internet topology as follows: we estimate LAP's latency based on the actual Round-Trip-Time of receiving HTTP packet headers and the estimated latency overhead of LAP cryptographic operations. For Tor, we measure latency using the actual Tor network. Specifically, we measure the latency with and without Tor between 10 geographically distributed machines and the top 200 university websites reported by Alexa[8], and also resolve the URLs of these sites in advance to exclude DNS lookup time. We use university sites as they are less likely to redirect traffic based on source addresses (in contrast to popular commercial sites). As Figure 8(b) shows, LAP users experience significantly lower latency compared to Tor users: 90% of LAP requests finish in less than one second, while most ($> 99\%$) of Tor requests take more than one second.

B. Throughput Evaluation

We evaluate LAP's impact on throughput using Netperf 2.5.0[9] with synthetic traffic of different packet sizes. Figure 8(c) shows the average throughput of LAP-disabled and LAP-enabled systems. We observe that the throughput grows with packet size for both cases. In particular, the throughput for the LAP-enabled case is slightly lower than the one for LAP-disabled, since it takes more time for LAP to process a packet than to simply forward it. However, the difference in these throughput is very small or even negligible, especially when the packet size is beyond 1 KByte. This result confirms that LAP has a small impact on router performance.

We also compare the throughput between LAP and Tor using a small testbed that runs LAP as well as a private Tor network with three Tor relays. For this evaluation, we set four machines in the testbed to be connected among each other using 1-Gbps links, each machine dedicated to be a source, a destination (file server), an intermediate machine running three Tor relays, and a Tor directory server. With this testbed, we measure the average throughput of a client machine that is downloading a 10-GB file from the file server for LAP and Tor. When downloading a 10-GB file using the Tor network, the client's average throughput is $\mu = 50.79$ Mbit/s ($\sigma = 1.41$). With LAP, $\mu = 939.50$ Mbit/s ($\sigma = 32.76$), showing a significant throughput increase.

To summarize, the overhead that LAP imposes is minor, which makes LAP suitable for practical deployment. In particular, at the cost of a small throughput decrease, LAP can improve the anonymity in current IP networks.

IX. Related Work

The most closely related schemes for anonymity protection, namely Tor Instead of IP [14] and AHP [7], are described and compared in the security analysis section (Section VII).

High-stretch anonymity systems. In Chaum's mix network [8], layer-encrypted messages are sent through a list of mixes, each of which can buffer, reorder, decrypt/encrypt these messages to defend against a global eavesdropper. However, delaying and reordering renders it impractical for real-time communication.

Onion routing systems, such as Tor [1], enable low-latency, bi-directional anonymous communication by sending layer-encrypted packets through indirect and unpredictable cryptographic circuits [37]. Unlike mix networks, onion routing systems are designed to defend against a local attacker (or a government-class attacker, as referred to in this paper) that observes only a fraction of the network. Under some realistic attacker scenarios, onion routing systems are shown to be more secure than mix networks [38]. Tarzan [39] explores onion routing in a peer-to-peer setting, and ANDaNA [40] adopts Tor in content-centric networking. However, onion routing systems still suffer from high latency due to high path stretch. To reduce Tor's latency,

[8]http://www.alexa.com/topsites/

[9]http://www.netperf.org/netperf/

new relay selection algorithms are suggested considering relay geolocations or link characteristics in addition to relay bandwidth [4], [41]. However, further studies are required to understand their impact on existing attacks against Tor.

Researchers have also explored solutions without layered encryption. For example, Information slicing [42] achieves source and destination anonymity through multi-path and secret sharing. However, Information slicing operates on overlays and suffers from noticeable latency. Crowds [43] leverages a crowd of users to collaboratively remove the trace of the real requester, and Hordes [44] exploit the inherent crowds within muticast groups for receiver anonymity. However, both Crowds and Hordes significantly stretch end-to-end paths.

Low-stretch anonymity systems. Using a single anonymous proxy such as anonymizer.com [45] results in low path stretch. However, users have to trust a remote proxy in burying the linkage between a sender and a receiver, and the proxy could easily become a single point of failure.

Censorship-resilient systems such as Decoy routing [46], Telex [47], and Cirripede [48] rely on ISPs to redirect traffic to blocked destinations. Although they also require enlisting ISPs for protection as LAP does, they place trust on remote ISPs to help defend against a much stronger adversary monitoring local networks.

Attacks on anonymity systems. Several researchers have studied how to passively and actively attack anonymity systems. For passive attacks, the adversary attempts to de-anonymize traffic by observing side-channel information such as packet timing [49], clock skew [35], and unique system state [50], [51]. However, such passive attacks often fail to scale or rely on information leaked from higher layer protocols. On the other hand, active attacks can accelerate traffic correlation. DoS is one type of active attacks that can be used for additional attack opportunities [32]. For example, by clogging the network and monitoring the latency change, the attacker can identify Tor entry nodes [20], [21] and locate Tor users [22]. Although our main objective is to camouflage one's topological location to enhance anonymity and privacy, LAP can mitigate DoS-based attacks by selectively publishing encrypted paths.

Low-latency anonymity systems are shown to be inherently vulnerable to timing and traffic analysis [22], [23], [52], because an adversary can easily correlate the traffic patterns of a sender and a receiver. Since our goal in this paper is to provide topological anonymity, we consider such temporal side-channel attacks as future work.

X. Conclusions

Current anonymous communication systems achieve a high level of anonymity against a strong attacker model, but pay a dear price in terms of overhead: high communication latency with high in-network computation and storage state. Especially the high latency causes the Internet browsing experience to endure a significant slowdown.

Anonymous communication would thus be more usable with reduced overhead. Indeed, we believe that many users can live with a relaxed attacker model, as they can trust their local ISPs but want protection from tracking by ISPs that are further away (potentially in other countries with different privacy laws) and from tracking by websites. Given such a weaker attacker model, we attempt to provide source and destination anonymous communication, session unlinkability, and location privacy at a very low overhead, barely more than non-anonymous communication.

In this framework, our approach is simple yet effective: by leveraging encrypted packet-carried forwarding state, ISPs that support our protocol can efficiently forward packets towards the destination, where each encrypted ISP-hop further camouflages the source or destination address or its location.

Although encrypted packet-carried forwarding state is currently not supported in IP, we design simple extensions to IP that could enable this technology. In particular, our approach is even more relevant in future network architectures, where the design can be readily incorporated.

This new point in the design space of anonymity protocols could also be used in concert with other techniques, for example in conjunction with Tor to prevent one Tor node from learning its successor. Despite weaker security properties than Tor, we suspect that LAP contributes a significant benefit towards providing topological anonymity, as LAP is practical to use for all communication.

Acknowledgements

We gratefully thank Soo Bum Lee and Sangjae Yoo for their help with Tor experiments, Nicholas Hopper and Paul Syverson for bringing related work to our attention and providing insightful feedback, and the anonymous reviewers for their valuable comments.

This research was supported by CyLab at Carnegie Mellon under grants DAAD19-02-1-0389 and W911NF-09-1-0273, from the Army Research Office, and by support from NSF under the TRUST STC award CCF-0424422, CNS-1040801, CNS-1040735, and CNS-0845896. The views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of ARO, CMU, NSF or the U.S. Government or any of its agencies.

References

[1] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: the second-generation onion router," in *Proceedings of conference on USENIX Security Symposium*, 2004.

[2] J. A. Muir and P. C. V. Oorschot, "Internet geolocation: Evasion and counterevasion," *ACM Comput. Surv.*, vol. 42, pp. 4:1–4:23, December 2009.

[3] R. Dingledine and S. J. Murdoch, "Performance improvements on Tor — or, why Tor is slow and what we're going to do about it," 2009. [Online]. Available: https://www.torproject.org/press/presskit/2009-03-11-performance.pdf

[4] A. Panchenko, L. Pimenidis, and J. Renner, "Performance analysis of anonymous communication channels provided by Tor," in *Proceedings of Availability, Reliability and Security*, 2008.

[5] P. G. Leon, B. Ur, R. Balebako, L. F. Cranor, R. Shay, and Y. Wang, "Why johnny cant opt out: A usability evaluation of tools to limit online behavioral advertising," in *Proceedings of CHI*, 2012.

[6] R. Kohavi and R. Longbotham, "Online experiments: Lessons learned," *Computer*, vol. 40, pp. 103–105, 2007.

[7] B. Raghavan, T. Kohno, A. C. Snoeren, and D. Wetherall, "Enlisting ISPs to improve online privacy: IP address mixing by default," in *Proceedings of PETS*, 2009.

[8] D. L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Commun. ACM*, vol. 24, pp. 84–90, February 1981.

[9] 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 IEEE Symposium on Security and Privacy*, 2011.

[10] "MobilityFirst future internet architecture project." [Online]. Available: http://mobilityfirst.winlab.rutgers.edu/

[11] J. Naous, M. Walfish, A. Nicolosi, D. Mazires, M. Miller, and A. Seehra, "Verifying and enforcing network paths with icing," in *Proceedings of ACM CoNEXT*, 2011.

[12] A. Efrati, "'like' button follows web users," May 2011. [Online]. Available: http://online.wsj.com/article/SB10001424052748704281504576329441432995616.html

[13] "British telecom phorm pagesense external validation report," 2008. [Online]. Available: http://www.wikileaks.org/wiki/British_Telecom_Phorm_Page_Sense_External_Validation_report

[14] V. Liu, S. Han, A. Krishnamurthy, and T. Anderson, "Tor instead of IP," in *Proceedings of ACM Hotnets*, 2011.

[15] A. Pfitzmann and M. Köhntopp, "Anonymity, unobservability, and pseudeonymity — a proposal for terminology," in *Proceedings of PETS*, 2001.

[16] O. Berthold, A. Pfitzmann, and R. Standtke, "The disadvantages of free mix routes and how to overcome them," in *Proceedings of PETS*, 2001.

[17] P. Syverson, "Why I'm not an entropist," in *International Workshop on Security Protocols*. Springer-Verlag, LNCS, 2009, forthcoming.

[18] J. Krumm, "A survey of computational location privacy," *Personal Ubiquitous Comput.*, vol. 13, pp. 391–399, August 2009.

[19] S. Burnett, N. Feamster, and S. Vempala, "Chipping away at censorship firewalls with user-generated content," in *Proceedings of USENIX Security*, 2010.

[20] S. J. Murdoch and G. Danezis, "Low-cost traffic analysis of Tor," in *Proceedings of IEEE Symposium on Security and Privacy*, 2005.

[21] N. S. Evans, R. Dingledine, and C. Grothoff, "A practical congestion attack on Tor using long paths," in *Proceedings of USENIX security*, 2009.

[22] N. Hopper, E. Y. Vasserman, and E. Chan-Tin, "How much anonymity does network latency leak?" in *Proceedings of ACM CCS*, 2007.

[23] S. Chakravarty, A. Stavrou, and A. Keromytis, "Traffic analysis against low-latency anonymity networks using available bandwidth estimation," in *Proceedings of ESORICS*, 2010.

[24] M. TechNet, "DHCP best practices." [Online]. Available: http://technet.microsoft.com/en-us/library/cc780311(WS.10).aspx

[25] Z. Wang, Z. Qian, Q. Xu, Z. M. Mao, and M. Zhang, "An untold story of middleboxes in cellular networks," in *Proceedings of ACM SIGCOMM*, 2011.

[26] "The RouteViews project." [Online]. Available: http://www.routeviews.org

[27] L. R. Group, "Nearly 1.3 million add broadband in the first quarter of 2011." [Online]. Available: http://www.leichtmanresearch.com/press/051711release.pdf

[28] "Tor metrics portal: Users." [Online]. Available: https://metrics.torproject.org/users.html

[29] "Internet world stats." [Online]. Available: http://www.internetworldstats.com/

[30] J. Trostle, B. Way, H. Matsuoka, M. Tariq, J. Kempf, K. T., and R. Jain, "Cryptographically protected prexes for location privacy in IPv6," in *Proceedings of PETS*, 2004.

[31] H. Burch and B. Cheswick, "Tracing anonymous packets to their approximate source," in *Proceedings of LISA*, Dec. 2000.

[32] N. Borisov, G. Danezis, P. Mittal, and P. Tabriz, "Denial of service or denial of security?" in *Proceedings of ACM CCS*, 2007.

[33] A. Houmansadr and N. Borisov, "Swirl: A scalable watermark to detect correlated network flows," in *NDSS*, 2011.

[34] P. Mittal, A. Khurshid, J. Juen, M. Caesar, and N. Borisov, "Stealthy traffic analysis of low-latency anonymous communication using throughput fingerprinting," in *Proceedings of ACM CCS*, 2011.

[35] T. Kohno, A. Broido, and K. C. Claffy, "Remote physical device fingerprinting," *IEEE Trans. Dependable Secur. Comput.*, vol. 2, pp. 93–108, April 2005.

[36] "The click modular router project." [Online]. Available: http://read.cs.ucla.edu/click/

[37] P. Syverson, "A peel of onion," in *ACSAC*, 2011.

[38] ——, "Sleeping dogs lie in a bed of onions but wake when mixed," in *HotPETs*, 2011.

[39] M. J. Freedman and R. Morris, "Tarzan: a peer-to-peer anonymizing network layer," in *Proceedings of ACM CCS*, 2002.

[40] S. DiBenedetto, P. Gasti, G. Tsudik, and E. Uzun, "ANDaNA: Anonymous named data networking application," in *Proceedings of NDSS*, 2012.

[41] M. Sherr, M. Blaze, and B. T. Loo, "Scalable link-based relay selection for anonymous routing," in *Proceedings of PETS*, 2009.

[42] S. Katti, J. Cohen, and D. Katabi, "Information slicing: anonymity using unreliable overlays," in *Proceedings of NSDI*, 2007.

[43] M. K. Reiter and A. D. Rubin, "Crowds: anonymity for web transactions," *ACM Trans. Inf. Syst. Secur.*, vol. 1, November 1998.

[44] C. Shields and B. N. Levine, "A protocol for anonymous communication over the internet," in *Proceedings of ACM CCS*, 2000.

[45] "Anonymizer." [Online]. Available: http://www.anonymizer.com/

[46] J. Karlin, D. Ellard, A. W. Jackson, C. E. Jones, G. Lauer, D. P. Mankins, and W. T. Strayer, "Decoy routing: Toward unblockable internet communication," in *Proceedings of FOCI*, 2011.

[47] E. Wustrow, S. Wolchok, I. Goldberg, and J. A. Halderman, "Telex: Anticensorship in the network infrastructure," in *Proceedings of USENIX security*, 2011.

[48] A. Houmansadr, G. T. Nguyen, M. Caesar, and N. Borisov, "Cirripede: circumvention infrastructure using router redirection with plausible deniability," in *Proceedings of CCS*, 2011.

[49] B. N. Levine, M. K. Reiter, C. Wang, and M. K. Wright, "Timing attacks in low-latency mix-based systems," in *Proceedings of FC*, 2004.

[50] P. Eckersley, "How unique is your web browser?" in *Proceedings of PETS*, 2010.

[51] T.-F. Yen, Y. Xie, F. Yu, R. P. Yu, and M. Abadi, "Host Fingerprinting and Tracking on the Web: Privacy and Security Implications," in *Proceedings of NDSS*, 2012.

[52] S. J. Murdoch, "Hot or not: Revealing hidden services by their clock skew," in *Proceedings of ACM CCS*, 2006.