An Internet Architecture Based on the Principle of Least Privilege

Vincent Liu, Seungyeop Han, Adam Lerner, Arvind Krishnamurthy, Thomas Anderson
University of Washington

ABSTRACT

In this paper, we present a novel interdomain network architecture that is based on the application of the principle of least privilege. By applying this design principle from the ground up, we can reduce the scope for a large range of misbehaviors (both unintentional and intentional), including configuration errors, DoS attacks, malicious behavior by ISPs, traffic discrimination, and censorship.

We present a complete architecture that considers the control plane, name translation mechanisms and even business models. Our objective in this paper is to show that such a system is possible and what it takes to make it work. We show that in our system, users can trade off performance and privacy, with very little performance penalty for users who do not need additional security.

1. INTRODUCTION

Despite the Internet's critical importance, portions of its architecture are surprisingly fragile, and the network as a whole suffers under the weight of incessant attacks ranging from software exploits to denial-of-service (DoS). One of the main reasons for this state of affairs is that the Internet was designed for a benign and trustworthy environment, in which little consideration was given to security issues. This assumption is clearly no longer valid for today's Internet. Operational experience has uncovered numerous issues, and the list of known vulnerabilities includes:

- Enterprises, ISPs, and governments monitoring or discriminating against certain types of traffic.
- Prefix hijacks and malicious alteration or deletion of routing information from upstream networks.
- Byzantine errors by neighboring ISPs disrupting unrelated traffic (e.g., prefix disaggregation causing router crashes).
- Disruption of service by resource exhaustion attacks against network links and end hosts.
- Silent re-routing without source approval of route choices (e.g., rerouting of DARPA traffic to China).
- Divergence between the control and data plane, wherein ISPs claim that they have a route, but do not forward.
- ISPs disrupting traffic between other ISPs or endpoints (e.g., to gain commercial advantage),
- Compromised routers and router software bugs affecting other ISPs beyond the misbehaving one.

Industry and researchers alike have moved to counter these threats, yet there have been only modest gains to date, as the Internet's key vulnerabilities are deeply rooted in its architecture. For example, secure routing protocols such as Secure BGP can be used to verify the authenticity of BGP control traffic, but they do not limit the impact of compromised or faulty routers. Encrypting traffic can hide packet contents, but the Internet still discloses the packet header to every ISP along the path, enabling traffic discrimination and censorship. Laundering packets through an anonymizing overlay, such as Tor, is fundamentally limited as governments can blacklist Tor nodes or monitor Tor entry/exit traffic. Equally problematic is that router implementations have become complex, leaving much room for zero-day attacks.

Our goal is to provide high-performance, private, reliable, uncensorable communication as long as a working path exists. We take a radical approach: expose to network elements *the minimum information and control necessary* for the correct functioning of the network. By limiting the information and power granted to network elements, we reduce their scope for misbehavior. We also reduce the complexity of their implementations, making them less vulnerable to unexpected failure. For instance, transit ISPs only need to know the next/previous hop on the path and nothing else. Similarly, end hosts do not need the location of other end hosts or even the ability to directly contact them. Instead, all traffic in our system goes through ephemeral rendezvous nodes so that only explicitly requested traffic is delivered.

For packet forwarding, our system takes inspiration from work on robust routing [34], onion routing [10,38], and capability-based DoS resilience [11, 41]. Our contribution is to show that by strictly adhering to minimum information/control exposure, we can create a forwarding layer that covers a larger class of attacks, performs well, and is as simple or simpler than the above proposals.

However, packet forwarding is just a small part of what we consider to be the Internet. To fully realize an Internet architecture that is resilient to Byzantine behavior by both ISPs and users, we need to answer several additional questions: How do we disseminate routes? How do we retain economic incentives for forwarding? How do we design a name service that preserves anonymity and enhances DoS resilience? Is it possible to get similar performance as today's Internet? The rest of this paper addresses these issues.

Although radical, our architecture preserves the structure, flexibility, and efficiency of the Internet. We preserve the principle of the thin waist: transport and application protocols are layered over our protocol just as they operate atop IP today. Further, while ISPs no longer control global routing, they retain their business incentives and their ability to

define the terms of transit—which customers and peers they connect, and at what price. Instead, we put control over routing decisions in the hands of end hosts so that each can trade off performance and security according to their own needs.

In summary, we make the following contributions:

- *Architecture based on the principle of least privilege:* We introduce the first network architecture that minimizes the amount of information/control given to ISPs and other users. As a result, our architecture is resilient to faulty or Byzantine behavior while simultaneously simplifying ISPs, maintaining economic incentives, and providing performance/features similar to those of the current Internet.
- *Novel mechanisms for privacy-preserving payment and name translation:* Aside from delivering packets, we need additional mechanisms for payment and naming. We introduce two novel, Byzantine-resilient, and privacy-preserving mechanisms: one to maintain economic incentives for ISPs, and one to provide human-readable name translation.
- *Prototype and performance analysis:* To demonstrate that our approach is practical, we implement the proposed system and show that we can forward 38 Gbps using inexpensive hardware. We also show that routes in our system have similar latency compared to the current Internet while still preserving privacy.

2. MOTIVATION AND OVERVIEW

The high-level goal of our architecture is simple to state, yet challenging to realize: private, reliable, and uncensorable communication as long as a valid path exists. Specifically, if a path exists between two end hosts such that *(i)* every ISP on the path is willing to deliver traffic for their physical neighbors and *(ii)* not all of them are colluding, then their communication should be resilient to even Byzantine ISPs/users.

2.1 Least Privilege in the Internet

Our key observation is that each of the attacks listed above is because an ISP or user is able to access information or resources that are unrelated to their core purpose. For instance, censorship and traffic discrimination occur because ISPs can determine when packets are destined for a particular website or make use of a particular protocol—without this information, there is no basis upon which to discriminate. Similarly, routing problems occur because misconfigured or compromised routers can exert significant influence on the routing behavior of remote ISPs and users.

Our approach is to adhere to the principle of least privilege along with the following additional design principles:

- *Each network entity is given the minimum information and control required to do its job.* Minimal information removes any basis upon which to discriminate against traffic, thus limiting the failure model to uniform degradation of service. Minimal control reduces their scope for misbehavior and reduces their implementation complexity.
- *End hosts must be able to tune performance and security:* As a corollary to the above, information and control should stay with the hosts to whom the traffic belongs. Users can use this control to suit their particular needs, e.g., achieving performance similar to or better than that of today's Internet, or obtaining stronger security.
- *The architecture must preserve the structure and features of the Internet.* We owe the success of the Internet to more than just IP/BGP. For an Internet architecture to be practical, ISPs need incentives to carry traffic and end-users must be able to locate services using human-readable names.

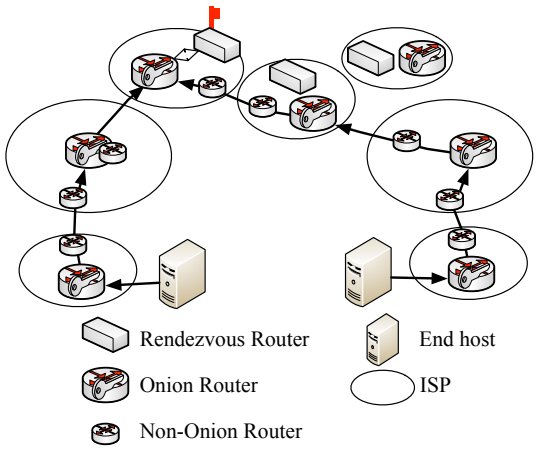

Figure 1: Overview of the entities involved in a single end-to-end connection.

Mechanism	Purpose	Section
Source routing	Minimizing ISP control, Defending against routing attacks	3.1
Onion routing	Preventing traffic discrimination, DoS resilience	3.1
RdV routing	Preventing traffic discrimination, DoS resilience	3.1
Credits	Economic model, Localizing failures, DoS resilience	3.2
Authorities	Trusted computing base for name resolution, Integrity of information	3.3
Single-Use Name Resolution	Network neutrality, DoS resilience	3.3
PoP-level Routing Vectors	Performance tuning	3.4

Table 1: Mechanisms used in our proposed architecture along with the section in which they are explained. Section 4.2 details their use in defending against attacks.

2.2 Baseline Design

We started our work by asking: what if the Internet used Tor at the network layer? Is such a system feasible, would it eliminate all of the mentioned attacks, and could it produce functionality/performance close to that of the current Internet? To be precise, we take Tor to mean the following:

- *Source Routing*, the ability for end hosts to choose routes.
- *Onion Routing*, the source host encrypting a packet with the key of each hop and having each hop iteratively unwrap the layers of encryption in order to hide the source, destination, and packet contents from intermediate hops.
- *Circuits*, an optimization that uses symmetric-key encryption over pre-established routes, rather than public-keys.
- *Rendezvous (RdV) Routers*, intermediaries to which both

endpoints establish a circuit in order to hide the source from the destination and vice versa, as in hidden services [1].

In such a system, ISPs would still be separate trust domains with their own equipment and internal structure, and would still develop their own business relationships and connections with adjacent ISPs. However, unlike the Internet, each ISP would also act as a single, logical onion router. End hosts would then incrementally build a circuit by negotiating a symmetric key and next hop with each ISP on the path.

ISPs would also optionally host RdV routers, which stitch together two circuits and forward packets from one circuit to another (i.e., source-RdV and RdV-destination). All end-to-end connections are then built by having both sides incrementally extend circuits through a sequence of ISPs to an ISP hosting a RdV router. Figure 1 depicts this process.

Strengths: The above proposal is attractive as it adheres to the principle of least privilege. For instance, ISPs are only aware of the previous and next hop, nothing else. In fact, this proposal already protects against a multitude of attacks including many types of traffic discrimination (as source, destination, and content are always hidden) and prefix hijacking attacks (as the source and destination jointly have full control over their routes). The principle of least privilege even provides some protection against DoS attacks. In particular, the indirection provided by RdV routers gives us the ability to use capability-based DoS protection as in i3 and Phalanx [11,41]. In other words, end hosts are not given the privilege to send directly to one another. Instead, the destination must explicitly reveal the location of a disposable RdV to the sender to grant it the capability to send.

What we have described is Tor at the network layer, which is an interesting idea in and of itself and the basis of a recent workshop paper [42]. Moving Tor into the network layer gives us the opportunity to address the issues of path dilation and the conspicuous nature of Tor overlay traffic.

Deficiencies: While this core idea forms the basis of our design, just using Tor at the network layer raises many issues.

- How do we compensate ISPs for transit? The consumer-provider/peering model of today's Internet does not apply to a source-routed architecture.
- Can we enable routing at the granularity of PoPs? Unlike the onion routers of Tor, ISPs that can potentially span entire countries. It is important for performance to be able to specify routes at a fine granularity.
- More generally, what does the control plane look like? Tor is an overlay protocol and was designed as such. To operate at the network layer, we need a completely different set of mechanisms for bootstrapping, routing advertisement and path computation, among other things.
- How do we provide human-readable names? This feature is essential to the Internet and has no counterpart in Tor.
- Are we actually able to provide performance similar to that of the Internet?

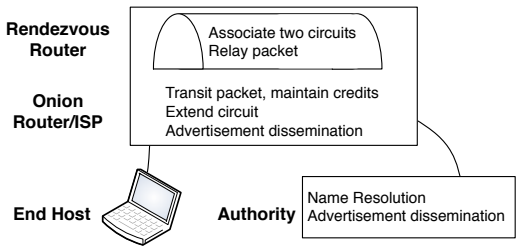

Figure 2: Key actors in our system and the functionalities they provide for end hosts.

- How do we do all of the above while continuing to adhere to the principle of least privilege?

2.3 Design Overview

In addition to the baseline design, we introduce additional mechanisms to address the above questions.

Credits: A novel form of digital currency whose transfer from one entity to another is a promise of payment. End hosts deposit them at each ISP on a circuit and use them to pay for circuit state and transfer of packets.

Authorities: Entities that provide name resolution and assist in routing advertisements. They control a distinct portion of the name space, based upon a hierarchical system of trust.

Single-Use Name Resolution: Name resolution maps a service name to a single-use RdV router.

Performance Annotated, PoP-level Routing Vectors: Routing vectors advertise, at a PoP-level, that a given ISP is willing to provide transit from an ingress ISP at a particular PoP to an egress ISP at a particular PoP.

Table 1 contains a slightly expanded list of the myriad mechanisms in our architecture, along with a brief description of what they contribute to security and fault tolerance.

3. ARCHITECTURE

There are four roles in our system: onion routers, rendezvous routers, authorities, and end hosts. The first three replace DNS, BGP, and the IP forwarding of today with a relatively simple interface (see Figure 2 for the functionalities provided). In this section, we describe our network architecture and how the first three roles combine to provide the desired interfaces. Later in Section 4, we provide examples of how end hosts and services can use these interfaces to defend against attacks while preserving flexibility and efficiency. In this section, we make use of the identifiers listed in Table 2 and the packet formats illustrated in Figure 3.

3.1 Forwarding

We begin the description of our architecture by outlining how packets are forwarded. At a high level, this follows the design illustrated in Section 2.2, with ISPs acting as onion routers, and end hosts setting up onion circuits to a target RdV router. The requirements for forwarding are as follows:

R1 Minimal information for ISPs: ISPs, including the one

Type	Bytes[1]	Scope	Usage	Generation
ISP	≥ 128	global	Self-certifying address, digital signatures, and encryption.	Every ISP generates a public/private key and uses the public key as an address.[2]
PoP	2	ISPs at PoP	Specifying routes, rendezvous addresses.	ISPs connected at the PoP negotiate an ID unique to each ISP. A single physical PoP can have many IDs to facilitate this.
Rendezvous	4	ISP-PoP pair	Ephemeral address that replaces IPs. Identifies a meeting point for 2 circuits.	For a new allocation, the ISP can assign an address in whatever manner they wish.
Hop ID	6	ISP-PoP pair	Identifies a circuit. Used to index into circuit state.	Chosen randomly by the extending ISP.

Table 2: Description of the types of identifiers found in packets.

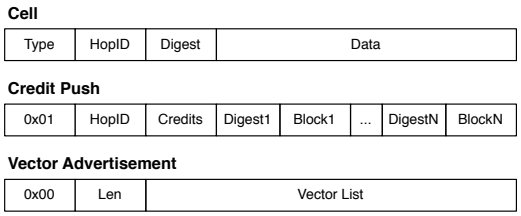

Figure 3: Packet formats used in our system.

hosting the RdV router, should know nothing more than the previous hop and the next hop. Giving minimal information to ISPs prevents traffic discrimination by removing any basis for it.

R2 Minimal control for ISPs: Similarly, ISPs should only have control over packets flowing though their own network. Limiting ISPs to this local scope prevents them from interfering with other ISPs and also reduces the complexity of their implementations.

R3 Minimal information/control for end hosts: The least privilege principle should also extend to end hosts. In particular, the two endpoints should be hidden from each other and should both have a say in their part of the path.

R4 DoS resilience: Finally, end hosts should be protected from DoS attacks from remote end hosts and ISPs. In fact, we argue that they should only receive packets when they explicitly request them.

We argue that a system with both source-routed onion circuits at the ISP-level and RdV routers satisfies all of these requirements. For instance, the primary purpose of both onion routing and RdV routers is to limit information exposure across the path. Thus these two mechanisms directly satisfy R1 and R3. Similarly, source routing takes all routing decisions out of the hands of the ISPs and thus provides R2. R4 is then provided by the fact that circuits need to be explicitly set up by an end host before packets can be forwarded to it. We elaborate on this proposal below. We omit detailed discussions of the mechanisms that are duplicated in Tor [10].

Packet Format. The primary packet type in our architecture is a fixed-length, onion-encrypted cell. This cell is one of three types of packets and is diagrammed in Figure 3.

[1]The number of bytes listed are just preliminary values used in our prototype. In particular, the ISP key can be variable size and use different techniques like elliptic curve crypto.

[2]An ISP is defined by its public key. Changing keys requires the ISP and its neighbors to rebroadcast all relevant advertisements.

They are used for both data and commands. In the case of data, the payload is always end-to-end encrypted so as to hide the contents even from RdVs. In the case of commands, onion routers know a cell is decrypted when the digest matches the data.

ISP Layout. ISPs store the following state for each circuit:

- *prev* = (HopID, PoP, ISP) tuple of downstream ISP or (HopID, EndHostID) if this is the first hop.
- *next* = (HopID, PoP, ISP) tuple of upstream ISP or (HopID, otherHopID) if this is the last hop.
- *symKey* established with the end host.
- *credits* available.

On top of storing this state, each ISP also exposes two forwarding-related interfaces to end hosts: circuit extension and packet transit. They also optionally provide an additional interface: connection to a RdV router.

Although we treat an ISP as a single *logical* onion router, this functionality can be distributed across multiple *physical* machines. For example, we could partition circuits across multiple routers inside a PoP using consistent hashing of *hopID*s. In the same way, if an ISP hosts or contracts out a set of RdVs in a given PoP, this functionality can also be distributed across multiple physical machines.

The low-level implementation of an ISP is out of the scope of this paper, but scalable circuit management is a well-proven technology widely used in the frontends of popular web sites.

Building a Circuit. Before data can be forwarded, an end host needs to first build a circuit. This happens through circuit extension, in which ISPs are iteratively appended to an end host's circuit until the circuit is finally attached to a RdV.

Circuit Extension: At a high level, end hosts initiate this process by sending an EXTEND_CIRCUIT command to the last ISP on the circuit (for a new circuit, this step is omitted). This message includes (a) the first half of a Diffie-Hellman key exchange, (b) the new ISP, T, and PoP to which the circuit should be extended and (c), the number of credits that the last hop should push to T.

The last ISP (or the end host for a new circuit) then sends a CREATE_CIRCUIT message to T. This message includes (a) and (c) from the extension request as well as a new, random *HopID*. At the end of this process, the end host will have negotiated a symmetric key with T, the last ISP on the circuit will have transferred (c) credits to T, and both ISPs will have updated their previous and next hop fields appropriately.

Attaching to a RdV router: After a circuit has been estab-

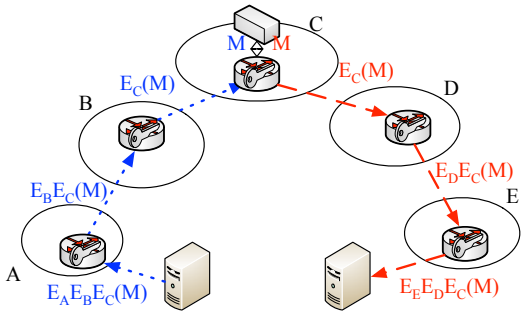

Figure 4: The encryption/decription at each hop of onion routing

lished to a target PoP, users can allocate a new RdV or connect to an existing RdV on-demand. Note that a final intra-ISP extension can be made if the RdV router is not in the ingress PoP (this does not incur another layer of onion encryption, but may require extra state). To allocate a new RdV router, an end host, Bob, sends a RENDEZVOUS_REQUEST command to the target ISP. This prompts the PoP to allocate a new, random RdV address and send it back to Bob.

Alice connects to Bob by building a circuit to the same target PoP, and sending a RENDEZVOUS_CONNECT request for Bob's RdV address. Alice can obtain the address either through a name resolution service (see Section 3.3) or out-of-band in the case of a P2P connection. The ISP then "grafts" the two circuits together such that the RdV forwards traffic between the two circuits bidirectionally.

Cell Forwarding. A connection is composed of two circuits grafted together by a RdV router. As a result, each packet goes through two phases: an upstream phase (toward the RdV router), and a downstream phase (away from the RdV router). The RdV router immediately and automatically forwards packets from either circuit to the other circuit.

After looking up the relevant circuit state with the cell's locally-unique *HopID*, the ISP strips one layer of onion encryption from each packet on the way upstream and adds a layer of encryption for each packet on the way downstream; however, as this is a symmetric key operation, both are operationally equivalent. Figure 4 shows an example of changes to a packet's layers of encryption. Note that in onion routing, when a packet reaches the RdV there will be no remaining onion-encryption, only the layer of end-to-end encryption.

This process terminates either when the packet reaches an end host or when the digest is correct, in which case the packet contains a command for the current hop. Data packets that reach the end of a circuit with no *next* hop are dropped.

3.2 Payment

Today's Internet is paid for with cascading payments from one ISP to another. End users pay their first-hop ISP for service, often for a maximum data rate or total amount of data. Each ISP will then, in turn, pay its provider for transit in both directions—normally on the basis of 95th percentile usage. This chain of payment proceeds from both ends of the connection until the path flows through a peering or provider→customer link.

Unfortunately, this economic model relies heavily on Internet's current structure and policies, and it is not compatible with source routing. Thus, we need a way for end hosts to send ISPs payment for packet transit and circuit state. Furthermore, the mechanism must satisfy the following:

- R1 Arbitrary Routes: It must be possible for end hosts to assemble an arbitrary route and funnel some form of payment to each and every ISP on the path.
- R2 Privacy-preserving: Payments should not leak any information about the end host or the path to remote ISPs.
- R3 Local transactions: ISPs should not need to take payment directly from remote ISPs or users. In the current Internet, money only changes hands between direct neighbors, to whom ISPs have business relationships.
- R4 Verifiable: End hosts should be able to verify that payments were made correctly. In the case of a faulty/malicious ISP, the end host should be able to pinpoint the point of failure so as to avoid that ISP in the future.
- R5 Expressive Pricing: First-hop ISPs must be able to provide a variety of different types of pricing plans. At a minimum, they should be able to approximate today's myriad pricing plans with credits.

To enable privacy-preserving payment for transit, we introduce a form of digital currency called *credits*, whose transfer from one entity to another is a promise of payment. By pushing credits from neighboring ISP to neighboring ISP, we can handle arbitrary paths as required by R1.

3.2.1 Economic Model

In the current Internet, payment is hierarchical. Tier 1 ISPs peer with each other, have no providers and therefore pay no one for transit. Instead, they are paid by others for use of their network. The cost is propagated downward to ISPs and users further down the hierarchy. Thus, the current Internet's payment schemes are opaque, implicit and assume much about policy and routing.

In our architecture, on the other hand, our goal is to make pricing transparent and explicit. Rather than implicitly including transit costs of remote ISPs in pricing, end hosts in our system pay ISPs for operation of their own network and no one else's. At the same time, we try to maintain the key features of the current Internet's economic model.

One example of an important feature we maintain is R3: the notion that connection between two ISPs is a result of a direct business relationship. Toward this end, ISPs in our system only ever promise payment to direct business partners. The exact conversion between credits and money is on a per-contract basis. In fact, they may still choose to engage in a peering agreement, where both ISPs agree to exchange traffic settlement free.

Credits are also flexible enough to satisfy R5. For instance, a first-hop ISP can implement per-month data caps by limiting the total number of credits that end hosts can push into

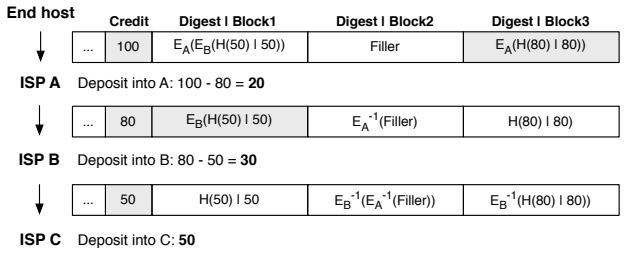

Figure 5: The journey of an example credit push packet. Shaded blocks indicate a block is relevant to the next hop.

the network. Similarly, bandwidth caps can be implemented by bounding the number of credits and packets an end host can push per time period.

3.2.2 Credit Push

Besides the ability to drop off credits along the circuit, the main requirement of our credit push mechanism is to maintain the privacy afforded by the rest of the architecture. For instance, Tor expends much effort to ensure packets look different at each hop and to keep intermediate nodes from discovering the length of the circuit. We take care to maintain the same guarantees here.

Packet Format. Credit push packets iteratively transfer batches of credits along a circuit. A single credit push message can deposit multiple credits to multiple hops along the circuit.

Toward this end, credit push packets contain a fixed number of credit blocks (as depicted in Figure 3). Each of these blocks is intended for at most one ISP, and includes the number of credits to push to the next hop, plus a digest of the value. The current number of credits being pushed from the previous hop is kept in the `Credits` field of the packet.

As an example, a block intended for ISP *A* with a value of 10 instructs *A* to transfer 10 of its credits to the next hop.

Preventing Correlation. At a high level, we treat each individual block as a mini, onion-encrypted cell. When an end host prepares a credit push packet, it encrypts each block with the chain of symmetric keys starting from the target ISP and ending with the first-hop ISP, as if the end host were individually onion routing each block to each target ISP.

Per-hop encryption gives us the property that the packet looks different at each hop. Without this property, two colluding ISPs on the path could see that they are handling the same circuit, thereby violating R2.

Hiding Length/Position in the Circuit. Furthermore, we prevent an ISP from knowing where it is located in the circuit by fixing the number of blocks in a packet and shuffling their order before sending. If there are fewer target ISPs on the path than blocks in the push message, the remainder are filled with random bits. If there are more, the end host must send multiple messages.

Protocol. An end host begins by crafting a packet, setting `Credits` to the number of credits to push to the first-hop ISP and generating a set of credit blocks as explained above.

Each ISP, upon receiving a credit push packet, deposits any credits from the previous hop. It then decrypts all the blocks with the *symKey* of the circuit and looks for a block whose digest matches its value. If it exists, the ISP simply changes the `Credits` field to the amount specified, deducts the same amount from the circuit and forwards the entire message to the next ISP in the circuit. After this process, every credit push block has one less layer of encryption, even if it was a filler block or already reached its destination.

As a simple example, consider the network in Figure 4. If A, B and C require 2, 3 and 5 credits per packet, the sender would need to craft a credit push cell containing two valid blocks, with 80 and 50 credits for A and B, respectively. The sender would then send the packet to A along with a promise for 100 credits. Upon receipt, A decrypts all N blocks and transfers the packet along with a promise for 80 credits to B. B decrypts all N blocks as well, and transfers 50 to C, who drops the packet because she is the last hop on the circuit. At the end of this process, A has 20 credits, B has 30, and C has 50, enough for 10 packets. Figure 5 illustrates this process.

Verification. End hosts can query the credit balance at any ISP in a circuit to verify deposits and deductions. If an ISP fails to forward credits or overcharges for service, end hosts can detect this misbehavior. They can even diagnose where packets are being dropped, as the first correct ISP will have a higher credit count than the upstream ISPs.

This verification technique is very powerful, but can only pinpoint to the granularity of a link. For instance, in the case where ISP A was supposed to forward 20 credits to ISP B: if A deducted the 20 credits and B did not add them, it is unclear who is lying. In this case, the end host either continues to use one of them until it misbehaves (if it does, we know the other one told the truth) or alternatively, stop using both.

3.2.3 Paying for Services

Both ends of every connection are responsible for pushing credits along their side of the connection. Two actions require payment: packet transit and circuit state. The price for both of these are advertised along with the routing advertisements described in Section 3.4.

In the case of packet transit, ISPs deduct the advertised amount credits as part of packet processing. In addition, the ISP periodically deducts credits for circuit state. As mentioned above, packet forwarding is best effort, so ISPs continue to forward packets and store the circuit state only as long as there are enough credits available.

Note that the RdV router does not need to deduct credits per packet, as the ISP is already getting paid for the state required and for every packet. Thus, the incentive for deploying RdVs is as a traffic attractor.

3.3 Name Resolution

Human-readable names are another essential service for the usability of the Internet. In the context of our architecture, we need to be able to map a service name to a RdV

address. Similar to the current Internet, we assume the existence of separate, hierarchical namespaces, each managed by an authority that is responsible for name resolution requests.

Sadly, a direct transplant of DNS is not sufficient. DNS essentially provides static mappings that, in our system, would allow ISPs to target specific services. For example, if a large ISP wished to target a service *S*, it could simply query for each of the RdV routers of *S*. It could then either DoS all known RdV routers or, if any of the RdV routers are within the ISP's own network, it could selectively block them.

Single-Use Records. We prevent adversaries from discovering the service associated with a given RdV router by making the mappings *single-use and unpredictable*. Every time a hostname is resolved, we return a new RdV address.

Compliant to the principle of least privilege, only the current client, target server, and the responsible authority know which service is using any particular rendezvous address. In this way, we guarantee that any RdV addresses that are compromised by adversaries are never used for real connections.

Hierarchy of Authorities. The authorities that provide these records are organized in a hierarchy where each is responsible for a disjoint subset of the namespace (e.g., foo.com is signed from a different authority than foo.bar.com or foo.ch). In many ways, authorities are like any other service—users connect to them through rendezvous routers, their network location is hidden, they are DoS-resilient, and users can find their rendezvous routers through other authorities.

In our system, there is a single root authority with a well-known public key.[3] Like in the current Internet, each ISP is responsible for obtaining RdV addresses for the root authority and providing them to users.

The root authority signs public keys and provides RdV address mappings for its child authorities. This process happens recursively to create a tree of authorities, each with a public key that can be traced back to the root.

Though there is a single root, sub-authorities are independently-managed. Therefore, embattled services can embed themselves within the hierarchy such that it would be impossible for the root authority to block them without blocking a large subset of the Internet.[4]

Protocol. Authorities provide mappings of the following format: $\langle name \rangle \rightarrow \langle PK_S,\ ISP{:}PoP{:}R \rangle$, where PK_S is the public key of the service in question and $\langle ISP{:}PoP{:}R \rangle$ is the address of one of many rendezvous routers for that service.

An end host requests name resolution by sending its public key, PK_{EH}, along with the name of the target service.[5] The response is of the form $E_{PK_{EH}}(\langle PK_S,\ ISP{:}PoP{:}R \rangle)$. Note that both the request and response are public-key encrypted.

[3]We use a single authority for simplicity. Extension to multiple such root authorities is straightforward.

[4]To take the example of Wikileaks in 2010, the only TLD willing to host their domain name was .ch, Switzerland. To block Wikileaks, the root DNS server would need to block all Swiss websites.

[5]They may also provide a target location to ensure locality as described in Section 4.3

Services that wish to have a human-readable name contract with an authority they trust to reserve a domain name and exchange public keys with them out-of-band. The service is then responsible for allocating rendezvous routers and sending them to the authority periodically.

Under high-load, an authority can request that users solve a computational challenge (as in [33]) before receiving a mapping. Users who solve more difficult puzzles receive service priority over others. However, this mechanism only activates during a DoS attack/high utilization so the common case will only require one round-trip latency.

3.4 Route Advertisements

In order for end hosts to be able to choose routes, they need to know which ISPs are providing transit as well as latency/cost of the routes. In this section, we describe route advertisements and how they are propagated.

PoP-level Routing. ISPs advertise *routing vectors*, which indicate that they are willing to provide transit from an ingress ISP at a particular PoP to an egress ISP at a particular PoP. Following the principle of least privilege, we grant ISPs no control over the segments chosen for a path, only over the segments from which an end host may choose. As in today's Internet, PoP-level routing enables better path choice based on geographical and topological properties.

Advertisement Format. Each routing vector consists of three ISPs (i.e., previous-hop ISP, advertising ISP and next-hop ISP), the two PoPs in the advertising ISP that connect them, and path information: $\langle$ISP$\rangle{:}\langle$PoP$\rangle{:}\langle$ISP$\rangle{:}\langle$PoP$\rangle{:}\langle$ISP$\rangle{:}info$. The last element may include information such as latency, cost, etc. that end hosts use to compose paths.

Advertisement packets are simply a signed list of these PoP-level routing vectors as illustrated in Figure 3. These packets need not be encrypted.

Advertisement Propagation. ISPs must disseminate and collect routing vectors as they expire and change. Whenever an ISP wishes to advertise a new route, it broadcasts routing vectors to all of its neighbors. Propagation is then straightforward – ISPs store and rebroadcast to their neighbors any new advertisements they receive from their other neighbors.

Note that we require authorities to collect route advertisements in addition to ISPs. By requiring that all ISPs and authorities be able to provide this information, we ensure that if an end host can reach *any* non-colluding ISP or authority, it can bootstrap the entire system.

End hosts obtain vectors by sending a GET_VECTORS command through an onion circuit to any ISP or authority, who then responds with the requested information. While end hosts get their first set from their first-hop ISP, they can later query any other ISP or any authority to cross-validate the set.

Note that routing vectors should change slowly and thus can be cached for long periods. They can also be compressed to reduce overheads (e.g., group similar advertisements). See Section 5.2 for an analysis of the size/churn of this set.

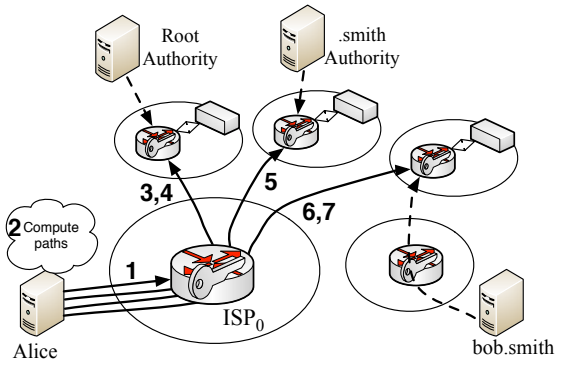

Figure 6: Network used in our end-to-end example. Dotted lines indicate pre-established circuits.

4. END-TO-END SYSTEM INTEGRATION

Unlike the current Internet, where a local problem (e.g., a misconfigured router) can cause a global outage, the routing functionalities described above are limited in scope and therefore limit the scope for misbehavior. we allow end hosts (both users and services) to maintain control over their own traffic and to customize behavior for their own particular needs. This flexibility allows many potential policies, but in this section, we provide several representative examples of how our architecture can be used to defend against the attacks listed in the introduction while preserving the flexibility and efficiency we have come to expect from the Internet.

4.1 End Host Operation

To tie our architecture together, we show a very simple case. An end host, Alice, wishes to connect to the network and send a cell to some service bob.smith. We assume services and authorities have already allocated RdV routers and given the relevant authorities their addresses. We also assume ISPs have collected the root's RdV addresses and all routing advertisements. Figure 6 depicts our simple example network and the following steps taken by Alice:

1. When Alice first connects to the network, her first-hop router sends a root authority rendezvous address and all of the routing advertisements it knows about.
2. Alice composes the routing advertisements to form a path to the root authority's rendezvous router, under the constraint that the path passes through more than one trust domain, but is otherwise the cheapest path.
3. She creates the circuit with a CREATE_CIRCUIT message followed by several EXTEND_CIRCUIT commands (one for each additional ISP on the path) and also pushes a batch of credits to each hop on the path.
4. Alice sends RENDEZVOUS_CONNECT command to connect to the authority and a GET_MAPPING command for the smith authority.
5. She receives both an address and public key for the smith authority and repeats the same process to find an address and public key for bob.smith.
6. She builds one final circuit to bob.smith's rendezvous router, and sends it a RENDEZVOUS_CONNECT.
7. Finally, Alice tunnels half of a Diffie-Hellman handshake across the circuit to bob.smith, who returns the favor to complete connection setup.

4.2 Defending Against Attacks

In Section 1, we list the vulnerabilities that are inherent to the Internet's architecture and that have plagued the system in recent years. Here we show how our architecture negates each of those threats and others, either implicitly or with slight modifications to the above base case.

Compromised routers, software bugs, etc. in byzantine remote ISPs. Our architecture stems from a few key ideas, foremost being the principle of least privilege. Functions of ISPs in our system operate independently of other network entities, and therefore failures are self-contained. The only malicious behavior available to ISPs is to either send too many (resource exhaustion) or too few (dropping) packets. Defense against the first is described below, and the second is easily detectable (via credits) and repaired.

Deletion or alteration of routing information. By design, every ISP and authority has a full set of routing information. Thus, end hosts can detect and recover from omissions by cross-validating information from multiple sources—if it can reach an ISP or authority that is not colluding with the attacker, it can fill in any missing information. Furthermore, alteration (e.g., prefix hijacking) is impossible since all advertisements are signed by self-certifying addresses.

Traffic monitoring and discrimination. Single-use rendezvous routers and onion routing ensures that ISPs are not aware of what rendezvous routers are used for or what type of information circuits/packets carry. Assuming the cryptography is strong enough, they have no way of monitoring traffic and no basis upon which to censor and discriminate against data or commands.

Resource exhaustion attacks. Large ISPs should be able to handle high traffic and are paid for every packet regardless. Small ISPs need not advertise any transit vectors, thus choosing to only serve direct customers and gaining the same protection given to end hosts in our system (i.e., adversaries cannot direct packets into the ISPs unless end hosts pay credits to receive the traffic).

For services, two properties protect them against resource exhaustion: anonymity and the requirement that all traffic be explicitly requested. Adversaries therefore do not know where to attack and cannot execute those attacks at a large scale. Attacks at the name resolution level are also difficult as adversaries only ever receive their fair share of addresses.

Censorship of a service. Resistance to traffic monitoring/discrimination and resource exhaustion attacks directly protect against censorship. They make it difficult to target services, their traffic, or their name mappings directly. Part of this is that adversaries do not know which packets/RdV

routers to attack. The other part is that adversaries are limited to their fair share of resources by computational puzzles. The service is even protected from higher-level authorities by the fact that embedding the authority deep within the hierarchy discourages blocking.

Silent rerouting. End hosts in our architecture perform source routing, and therefore choose their own paths. In fact, transit ISPs do not know where circuits are coming from nor where they are going. Hence, there is no basis upon which to selectively reroute certain types of traffic, and any attempted rerouting will simply break the path.

Divergence between the control and data plane. Expected behavior of an ISP in our system is relatively simple—deliver packets through an advertised routing vector for the advertised price. This simple contract gives our architecture a robust way to pinpoint failures and misbehavior.

End hosts can, at any time, ping an ISP on one of its circuits to check the remaining credit balance. The credit counts paired with ISP cost advertisements essentially provide a detailed, per-hop packet count that can be used to verify correct transfers and charges. This metric can be used to pinpoint the link at which the failure occurred.

Deanonymization. The security of onion routing requires the circuit to traverse multiple entities who are not all colluding. In some cases, particularly when the rendezvous router and end host are close, the shortest path may remain within the jurisdiction of a single government. One solution is for end hosts to use real-world information to construct paths, e.g., require paths to pass through three different countries.

Anonymity is preserved even in the extreme case where an ISP prevents its customers from connecting to remote rendezvous routers. Though users' circuits stay within the ISP, the other half of the circuit will continue to hide any actionable information. Services can remain anonymous as well by laundering packets through a proxy outside of the ISP. Note that this attack disrupts approved usage as well since it precludes interoperability between other ISPs with the same restriction and prevents usage of smaller services that have limited rendezvous router coverage.

State Correlation. Onion routing ensures that packets and circuit state are different at each hop. We add to the circuit state a record of the credits remaining, which if used naively, can leak information about a circuit's length or destination. End hosts can defend against this attack by giving a random number of extra credits to certain hops in the circuit to imitate a longer path. Credits are worth a very small amount of money so leaving them in the middle of the network is only marginally more expensive than the common case.

4.3 Preserving Flexibility and Performance

As a side effect of granting more control to end hosts, we are able to preserve (and in some cases improve) the flexibility and efficiency we have come to expect from the Internet. In this section, we describe how end hosts can trade off performance and security according to their own needs. Section 5 evaluates some of these techniques.

Latency. For end hosts that do not require additional anonymity, latencies in our system can often be less than they are in the current Internet. Instead of relying on ISP policies and BGP, end hosts in our system are free to chose the fastest path because they have full control over paths as well as access to latency information included in routing vector.

Further, services in large ISPs can choose RdV routers in the their home ISP so that any shortest path from a user to a RdV router is also part of a shortest end-to-end path. This also has the benefit that allocating new RdV routers is fast for the service as it does not require any circuit extensions.

Connection setup time. The latency penalty for setting up a circuit can be minimized by prefetching partial paths. The first few hops of each circuit are usually predictable since we expect large tier 1 ISPs to host most RdV routers. In this way, end hosts only need to set up the final hops on-demand.

Transport- and application-layer protocols. What we describe in this paper is a network layer that provides security by construction. This network layer preserves the principle of the thin waist and exposes to higher layers the abstraction of reliable, privacy-preserving forwarding. Like IP, we can layer higher-level protocols on top of our network layer.

Content delivery networks. Our architecture can also emulate techniques like CDNs, where servers placed close to users cache information to reduce latency and load on the network while increasing availability.

In our system, servers can be placed in exactly the same locations within ISPs. They would connect to a nearby RdV (perhaps in the same room), which would enable end hosts to use paths similar to today's. End hosts can then optionally provide a nearby PoP to the authority when requesting name resolution, who will then determine an appropriate RdV. Of course, end hosts can also choose to lie and use CDN servers far away or take roundabout paths for privacy reasons.

End host policies for choosing routes. End hosts in our system have full control over the route to the destination. To increase usability, we can provide pre-defined policies that automatically find the best route. These policies might include a preference for lowest latency within a cost limit or lowest cost within a latency limit. In addition, they might also specify the minimum number of ISPs to traverse to ensure security or even require that all paths pass through a specific ISP. We can efficiently calculate all of the above by varying the shortest path algorithm using the latency/cost graph of the network.

Caching There is also a slight performance penalty related to the inability to cache single-use name resolution entries at the ISP level. Caching can, however, occur at the end host (in the OS for example). To make actual name resolution accesses faster, we use the fact that each ISP has already set up a path to each root authority. An end host can therefore utilize prefetching techniques to have some path to

an arbitrary ISP, implying a prefetched path to any desired root authority. If the name resolution request requires recursive accesses to sub-authorities, we handle these in a similar way to normal prefetching. Unless the sub-authority is used enough to warrant a permanent prefetched path from either the ISP or the end host, we assume that most sub-authority rendezvous mailboxes are in the core and can therefore utilize the prefetching method outlined in the previous paragraph.

4.4 Example: Path Selection

We next explore a specific example of how end hosts might tune our architecture in practice. In particular, we focus on the problem of path selection, where an end host must efficiently choose from a set of routing vector advertisements to construct a path to a target location.

We consider three metrics when selecting paths:

- *Security* measured in terms of the number of distinct ISPs traversed by the packet
- *Performance* measured in terms of latency of the path from the source to the rendezvous router
- *Cost* in terms of the payments to be made to ISPs

In particular, we consider a constrained setting where the payments for traversing routing vectors are discrete integral values; in other words, costs are binned into different price levels, each of which can be expressed as an integer value. One example of this would be to attribute a zero cost to routing vectors that are policy compliant in today's Internet and to attribute a cost of one unit to those that are non-policy compliant (e.g., packet is received from a peer and transited to another peer).

Our candidate algorithm considers paths secure if they pass through at least k separate ISPs and insecure if they are shorter. A real path which used only ISPs governed by a totalitarian government might not actually be secure, as those ISPs would be likely to collude by government mandate. In such a case, a security metric which incorporated out-of-band information about legal and political jurisdictions would be required. Lacking that jurisdictional information, our path selection evaluation does not consider such a metric.

An end host can have simple or complex preferences. It might always prefer faster paths or always prefer cheaper paths. It may even require secure paths in general, but allow insecure routes in favor of performance when playing an online game or in favor of cost while streaming non-interactive media.

In Section 5, we consider an end host which always requires security; however, among secure paths, our end host always prefers cheaper paths, breaking ties between paths with equal cost by taking the lower latency path. We express the path selection algorithm as a shortest path computation over a transformed PoP-level graph. In particular, we develop an algorithm that has the following properties: (a) the path from the source to the rendezvous router traverses at least k different ASes, (b) the path is a minimum cost path amongst all paths that traverse k ASes, and (c) it is the lowest latency route amongst all minimum cost paths that traverse k ASes.

Given an input graph $G = (V, E)$, where V is the set of PoPs in the Internet and E is the set of routing vectors. The goal is to compute a path from source PoP s to rendezvous PoP t. We construct a new graph $G' = (V', E')$ that is specified as follows. Let $V' = \{[v_{curr}, v_{prev}, k] | v_{curr}, v_{prev} \in V, (v_{prev}, v_{curr}) \in E\}$. In other words, the set of vertices in the new graph is expressed as three-tuples that correspond to a PoP in the input graph, a neighbor of the PoP in the input graph, and a hop count.

In addition, the edge set is expressed as follows. Let $\langle v_1, A, v_2 \rangle \in E$ be a routing vector that connects an ingress PoP v_1 to an egress PoP v_2 of the AS A.[6] Similarly, let $\langle v_2, B, v_3 \rangle \in E$. We add the following edge to E': $\langle [v_2, v_1, k], [v_3, v_2, k+1] \rangle$. Further, we represent the weight of this edge with a two-tuple cost value: (c, l), where c is the payment cost associated with the routing vector and l is its traversal latency as advertised by the ISP. A weight tuple (c_1, l_1) is greater than another weight tuple (c_2, l_2) if either $c_1 > c_2$ or $c_1 = c_2 \wedge l_1 > l_2$. Note that this formulation encodes the previous PoP traversed in order to reach the current PoP, keeps track of the number of ASes traversed, and expresses the edge traversal cost using both a cost and a performance metric.

With the above graph, our goal is to determine the lowest weight path from the source $\langle s, -, 0 \rangle$ to any of the target nodes of the form $\langle d, *, k \rangle$, where "*" is a wildcard and k should be greater than the security parameter regarding the number of ASes traversed (e.g., $k \geq 3$). To compute the lowest weight path, we simply use Djikstra's shortest path algorithm, with the additional detail that we do not generate all of the vertices in the graph G' a priori, but rather do so on demand. The algorithm terminates as soon as we reach any one of the feasible target nodes.

The above example is just one of the many kinds of route selection strategies that can be employed at the end host. We study the performance of paths computed using this algorithm in the evaluation section.

5. EVALUATION

In this section, we explore the feasibility and efficacy of our proposal by answering the following questions:

- What is the hardware cost incurred by our system?
- Can the performance be similar to that of today's Internet?
- What are the overheads associated with the mechanisms?
- What is the cost of varying levels of anonymity?

5.1 Implementation and Experimental Setup

We begin by describing our experimental setup, which includes a custom, line-speed, user-level packet processing en-

[6]For ease of exposition, we assume that the PoP numbering is unique across all ASes, even though that is not required of the design presented earlier.

gine; a Click prototype implementation; and simulations on realistic Internet topologies.

Onion Router Prototype To prove the feasibility of ISP onion routing, we have built a prototype onion router that can encrypt and forward 1500 Byte packets at 38.3 Gbps using a modestly equipped server. It is implemented as a Linux application using DPDK,[7] a framework that allows us to write fast packet processing applications. We evaluate our implementation on a Dell PowerEdge T620 server with Dual Intel Xeon E5-2670 processors and 4 Intel X520 SFP+ NICs.

When the onion router receives a cell, it looks up the hopID, decrypts the data with the corresponding encryption key, and forwards it if needed. Note that the lookup is simpler than it for an IP router as we only need a hash-table lookup, rather than longest-prefix matching. Processing multiple packets in batches (currently 32), and processes the above tasks for each individual packet. Encryption is with 128-bit AES using the symmetric cipher with OpenSSL 1.0.1.[8]

Click Implementation We also developed a prototype using Click in order to test end-to-end connection performance in emulated cluster settings. The prototype implements all aspects of the data plane functionality, including circuit establishment, cell forwarding/credits, and rendezvous routers using Click [31]. We have written 23 Click elements, totaling over 4500 lines of code. These perform header validation/parsing, cryptographic actions, and routing table management. Cryptographic methods are implemented using OpenSSL with 1024-bit RSA for the public key cipher and 128-bit AES with CTR mode for the symmetric cipher. We deployed this prototype on the Emulab [45] testbed with user-mode Click. These nodes were connected in a linear topology of up to 8 nodes by links shaped to 100 Mbps bandwidth and 10 ms latencies.

Simulation Dataset To test global or Internet-scale properties of our architecture, we rely on simulation results using realistic Internet topologies. Specifically, we use the network topology and routing measurements collected by the iPlane project. The iPlane network atlas is built using measurements from over 300 PlanetLab sites to almost every routable prefix. It is augmented with reverse path measurements back from these prefixes in order to overcome biases introduced by the homogeneity of network paths seen from PlanetLab nodes [21]. The resulting network topology contains more AS-level links than other datasets such as CAIDA's AS level topology dataset [9] and the IXP mapping projects [17]. We performed our measurements using an atlas that comprised of 241,292 PoPs (where each PoP is a set of routers from a single AS colocated at a given geographic location) and 1,055,313 PoP-level links. We used PoPs at all tier 1s and most large tier 2s as locations where rendezvous routers can be hosted. This amounted to 387 PoPs spread across 65 ASes. We also processed the iPlane dataset to derive the set of routing vectors seen on actual routes through the Internet. This characterizes the business relationships between the various ASes, and we use this in constructing valid paths in our proposed architecture.

[7] http://dpdk.org/

[8] This version of OpenSSL utilizes AES-NI [25].

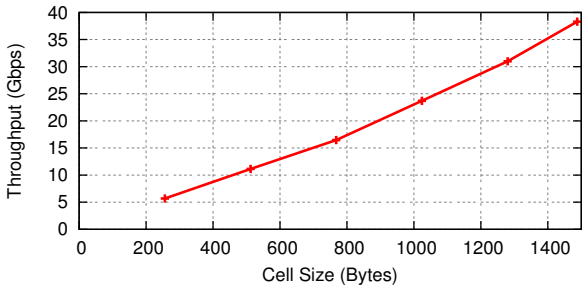

Figure 7: Throughput of our onion router prototype vs. cell size.

5.2 Feasibility

Though resistance to various faults come at the cost of increased hardware and computation, these costs are small compared to the potential benefits. In particular, we show:

- ISPs only need a small amount of extra hardware (which would be offset by the complete removal of security-oriented middleboxes and the simplification of routers).
- For end hosts that do not need extra anonymity, the performance is similar to that of the current Internet.

Throughput: We first ask whether or not onion encryption of each packet on the Internet is feasible and what ISPs would need to do it at line rate. Recent research suggests a positive answer to this question, as [25] showed 10 Gbps using DPDK, and [15] showed up to 20 GBps using GPUs.

In this paper, we test the throughput of a typical, relatively inexpensive server. We used the onion router prototype and measured its throughput with various packet sizes and with 10M concurrent onion circuits to index into. Figure 7 shows that the onion router throughput gets up to 38.3 Gbps.

A back-of-the-envelope calculation of what this means for a large ISP like AT&T tells us that the hardware cost is low. Using the worst-case projected peak usage of the biggest ISPs (15Tbps in 2011 with a growth rate of 45%/yr [26]) and AT&Ts PoP count [29] (around 120), we can estimate that just one of these machines per PoP is sufficient.

Pkt size	3 Hops	4 Hops	5 Hops	6 Hops	7 Hops
64	63.6	84.9	106.1	127.5	148.2
128	63.7	85.1	106.4	127.6	148.7
256	64.1	85.5	106.9	128.5	149.6
512	64.7	86.4	108.1	129.7	151.0

Table 3: Round-trip latency (ms) over an N hop circuit

Latency: We measured the latency overhead of our onion routing operations as a function of packet size and hop count. Experiments were over pre-established circuits on our user-mode Click implementation. Though our implementation was unoptimized, there are still several takeaways we can glean from these numbers. We varied path length between three to

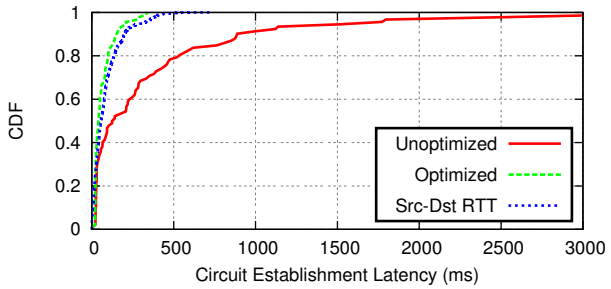

Figure 8: Circuit establishment latencies with and without optimization compared to the base end-to-end RTT. Optimization here indicates prefetching of partial paths.

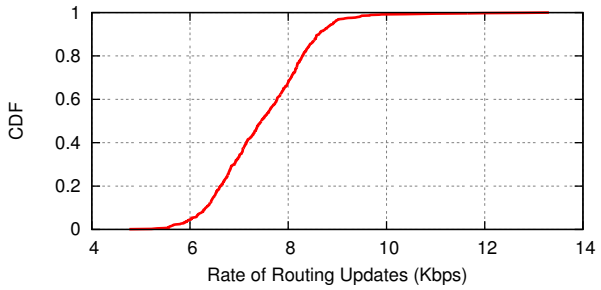

Figure 9: CDF of the rate of routing updates (in Kbps) for our architecture over a period of about 200 hours.

seven router hops (see Table 3) and measured the round-trip time from the client to the final router on the circuit.

Note that network delay dominates these latencies, not packet processing: at the worst, our system adds 8% overhead, or about 1.6 ms per hop. Further, most of the increased latency is not due to onion routing—a basic Click configuration which forwards traffic across the same paths without performing any processing produces similar latencies.

Circuit Establishment Latency: We next measure the circuit establishment costs incurred by our proposal. Recall that the destination providing a service establishes both a rendezvous router and a circuit to it before publishing the rendezvous router. In the unoptimized case, an end host desiring communication with the service would setup a circuit by incrementally extending a partial circuit one AS hop at a time until it reaches the AS holding the rendezvous router. Figure 8 shows that this circuit establishment cost could be substantial. As mentioned earlier, an end host could pre-establish a partial circuit to some ISP and then extend it appropriately to the ISP containing the rendezvous router upon receiving the identity of the rendezvous router. Figure 8 illustrates that this optimization provides significant gains, with the resulting circuit establishment latency comparable to the typical end-to-end latency in our Internet atlas.

Routing Update Frequency: Finally, we look at the feasibility of source routing using our routing vectors. To do so, we tracked routes over a period of about 200 hours to measure the churn of routes. Every 15mins, a multitude of PlanetLab nodes would traceroute and reverse traceroute 1500 prefixes. We used iPlane data to map the router addresses to

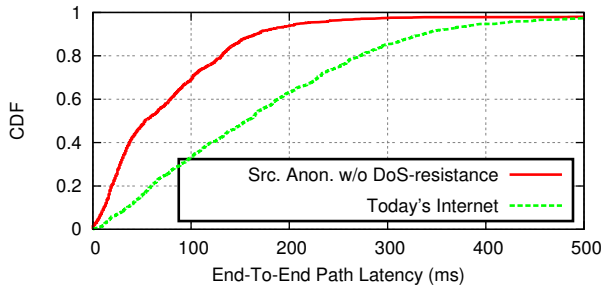

Figure 10: Comparison between latencies of end-to-end paths in today's Internet and a baseline user in our architecture that just needs source anonymity.

ASes, and then inferred routing vectors from those traces. By tracking these routing vectors over time, we are able to approximate the rate of routing vector updates in our architecture. Note that our results are a loose upper bound, as we do not compress/aggregate any of these vectors; additionally, routing updates in our system only happen upon permanent topology changes, not temporary changes in AS-level routing preferences like in the current Internet.

Figure 9 shows a CDF of the average rate of routing updates. Our results show that even with uncompressed updates that include headers and latency information, end hosts and ISPs only need to expend 8.6Kbps in the 90th-percentile.

5.3 Cost of Anonymity

In the following experiments, we take the role of an end host trying to access a given destination. We evaluate the performance of different levels of anonymity and privacy, and we consider them in contexts where paths are constrained to consist of routing vectors used in today's Internet.

Source Anonymity: As a baseline, we consider a user that requires basic anonymity, i.e., source (and not necessarily destination) anonymity. In particular, users can attempt to achieve source anonymity by requiring each path to pass through at least three ISPs, but otherwise choose the shortest latency path that is also policy-compliant with the transit/peering policies in practice today. On the other hand, if the destination does not care about its own privacy, then the ideal course of action, from a performance perspective, would be to co-locate rendezvous routers and servers in the same ISP. While this would not provide anonymity for the destinations and would require the destination's ISP to be capable of handling potential DoS attacks, it may be a feasible approach for a large service like Google.

Figure 10 shows a CDF of the latency for this use case and compares it against latencies in today's Internet, which we estimate through iPlane path predictions for the same source-destination pairs. Even though the source is constraining the path, *we still outperform the current Internet's BGP-based path selection*. This is due to two reasons: (a) the ability to perform source routing enables greater path diversity, and (b) by incorporating inter-PoP latency information, one can have more fine-grained estimates of performance

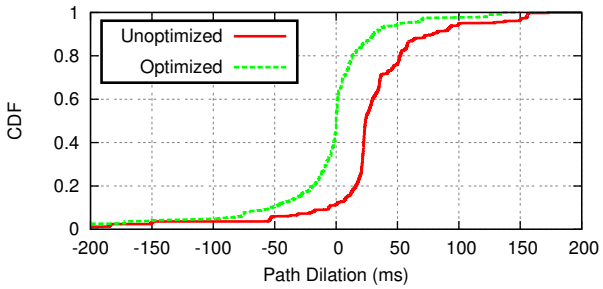

Figure 11: Path dilation when we need source/destination anonymity and DoS resilience. We show both optimized and unoptimized rendezvous router negotiation policies.

than using AS path lengths. Furthermore, we found that the anonymity constraint did not negatively impact most paths. Most of the shortest paths we observed already included 3 ASes—the restriction only affected 2.4% of paths. In other words, the path dilation of our architecture is small even compared to a pure source-routed scheme.

Censorship-resistant Web Services: We next evaluate what might be a web service's situation where the destination is concerned with both anonymity, traffic discrimination, and cost. Just like the source, the destination can ensure that its path to the rendezvous router passes through at least three ASes. It can also negotiate a set of rendezvous routers with each source to provide path diversity for protection against DoS attacks. Finally, we constrain routes to the rendezvous routers to be policy-compliant given the inter-AS relationships in place today. In other words, these routes would correspond to paths that would require the lowest (zero) credit payments from end-hosts to ISPs and would therefore make less use of path diversity compared to using higher cost paths. We consider two policies in which:

- *(Unoptimized)* the authority chooses a random rendezvous router through which to route.
- *(Optimized)* the destination provides to the authority a set of rendezvous routers that are acceptable—far enough away to allow for 3-AS paths and close enough to not incur too much path dilation—and the authority institutes a similar policy to choose a RdV given the source's nearest PoP.

Figure 11 compares the path dilation incurred by these two policies when compared to source routing through today's Internet. In the first, unoptimized case, latencies are increased by about 25 ms in the median case and about 100 ms in the 90th percentile, when compared against the source-anonymity-only case detailed above. In the second, optimized version of path selection, wherein the destination provides n rendezvous routers through which it is willing to receive packets, and the source picks m out of those (for our purposes, we set m:n to 1:6). We observe that the path dilation in the 90th percentile for this policy is only 25 ms, which could be viewed as an acceptable cost for defending against censorship, traffic discrimination, and DoS.

6. RELATED WORK

Improving Internet security has been a long-standing goal of the network research community. Our work benefits and borrows ideas from many different sources.

An early and inspirational approach was Perlman's Byzantine Routing protocol [34, 35]. She observed that a network with source routing and dedicated buffering for each source in every router is resilient to the misbehavior of an arbitrary number of routers. As long as some well-behaved path exists between the source and destination, the source will eventually be able to deliver a packet. Subsequent work has continued to explore the reliability and security of routing and forwarding [3, 6, 13, 19, 23, 32, 37, 48, 50].

These systems each solve different problems and take different approaches, but they all have a common vulnerability: sources and destinations are still visible to intermediate ISPs. This is fine if we can find a well-behaved path of ISPs, but in cases where there is no such path, e.g., when a first-hop ISP discriminates against traffic, there is no recourse.

The other attack to which the above proposals are vulnerable is Denial-of-Service, which has received a lot of recent attention [4, 11, 27, 41, 44, 47, 49]. Most similar to our work is Phalanx and i3, in which packets are first sent to a "mailbox" rather than directly to end hosts. In our work, rather than buffer packets in mailboxes until they are picked up by recipients with a put/get model, we provide the abstraction of a seamless connection. Further, we simplify filtering of extra packets—rather than provide a cryptographic token for each packet, end hosts grant capabilities simply by establishing an onion circuit to a RdV router.

Another rich field of research has been in source and destination confidentiality for the purpose of preventing snooping and discrimination [7, 8, 16, 18, 24, 36, 38]. Unfortunately, anonymity by itself is not enough to defend against attacks observed in the wild. In particular, resource exhaustion becomes easier in a world with strong privacy, as anonymity precludes a class of defenses against DoS attacks.

Recent research [33, 40, 43] has shown that there is a way to reconcile DoS and the existence of Sybils [12]. This research has used bandwidth and computation as resource proofs to prevent a single machine from acting like many machines. These solutions fundamentally have suboptimal performance. Our solution introduces the concept of money as a resource proof since it incurs no performance penalties and is perfectly-suited to the economic model of the Internet.

Another common approach for reliability and privacy has been to use an overlay network [2, 10, 39]. These systems are often better than the alternative, but they are fundamentally limited by relying on an unreliable and insecure substrate. Tor in particular has gained much popularity, but faces great challenges as a result of being an overlay. For instance, Tor necessarily incurs large path dilation as a result of bouncing through end hosts. Even distribution of source/binaries to censored countries is a challenge in practice.

Our routing vectors are reminiscent of pathlets [14], par-

ticularly their local transit policies. These are essentially a specialized version that provides us with the greatest level of privacy and end-host control, given an ISP's willingness to carry traffic from one neighbor to another. Though they are similar in mechanism and spirit, our focus on privacy and end host control require us to extend them to include finer-grained control, performance annotations and cost.

Previous work has also looked at pricing for inter-domain networks [22, 28]. However, they mostly focus on the market effects of pricing on resource management, whereas our main goal is a privacy-preserving way to pay for transit. One example is re-feedback [5], which requires that packets carry credits that are deducted at routers and cause drops when in debt. However, the goal of their mechanism is to expose congestion to the entire path—our goal is the exact opposite.

Finally, work has gone toward bypassing adversarial first-hop ISPs using steganography [20, 30, 46]. This is complementary to our work, and may be useful in a partial deployment scenario, where only a subset of ISPs participate.

7. CONCLUSION

The Internet has become an essential piece of society's infrastructure; without a functioning, reliable network, most modern systems would become nearly useless. Yet the Internet's architecture is not up to the task: it has numerous well-known vulnerabilities, and there has been only very slow progress to reducing its susceptibility to attack.

In this paper, we describe a new interdomain network architecture to achieve robustness, attack resilience, and privacy. By carefully limiting the role of network participants, we reduce the scope for misconfigured or malicious ISPs and endpoints to disrupt and snoop third party traffic. We show that our security architecture is consistent with ISP incentives and with end-to-end path performance.

8. REFERENCES

[1] Tor: Hidden service protocol. `https://www.torproject.org/docs/hidden-services.html.en`.

[2] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient overlay networks. In *SOSP*, 2001.

[3] T. Anderson, L. Peterson, S. Shenker, and J. Turner. Overcoming the Internet Impasse through Virtualization. *IEEE Computer*, 2005.

[4] H. Ballani, Y. Chawathe, S. Ratnasamy, T. Roscoe, and S. Shenker. Off by default! In *HotNets-IV*, 2005.

[5] B. Briscoe. Internet: Fairer is faster. Technical Report TR-CXR9-2009-001, May 2009.

[6] J. Chase, A. Yumerefendi, I. Baldine, Y. Xin, A. Mandal, C. Heerman, and D. Irwin. Cloud Network Infrastructure as a Service: An Exercise in Multi-Domain Orchestration. Duke University, 2010.

[7] D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, pages 65–75, 1988.

[8] D. L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–90, 1981.

[9] X. Dimitropoulos, D. Krioukov, M. Fomenkov, B. Huffaker, Y. Hyun, kc claffy, and G. Riley. AS Relationships: Inference and Validation. *SIGCOMM CCR*, 37(1):29–40, 2007.

[10] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-Generation Onion Router. In *USENIX Security Symposium*, 2004.

[11] C. Dixon, T. Anderson, and A. Krishnamurthy. Phalanx: Withstanding Multimillion-Node Botnets. In *NSDI*, 2008.

[12] J. R. Douceur. The sybil attack. In *IPTPS*, 2001.

[13] D. Estrin, J. Mogul, G. Tsudik, and K. Anand. Visa protocols for controlling inter-organizational datagram flow. *IEEE Journal on Selected Areas in Communications*, 1988.

[14] P. B. Godfrey, I. Ganichev, S. Shenker, and I. Stoica. Pathlet routing. In *SIGCOMM*, 2009.

[15] S. Han, K. Jang, K. Park, and S. Moon. PacketShader: a GPU-accelerated software router. In *SIGCOMM*, 2010.

[16] S. Han, V. Liu, Q. Pu, S. Peter, T. Anderson, A. Krishnamurthy, and D. Wetherall. Expressive privacy control with pseudonyms. In *SIGCOMM*, 2013.

[17] Y. He, G. Siganos, M. Faloutsos, and S. Krishnamurthy. A Systematic Framework for Unearthing the Missing Links. In *NSDI*, 2007.

[18] H.-C. Hsiao, T. H.-J. Kim, A. Perrig, A. Yamada, S. C. Nelson, M. Gruteser, and W. Meng. LAP: Lightweight anonymity and privacy. In *IEEE Symposium on Security and Privacy*, 2012.

[19] Y.-C. Hu, A. Perrig, and M. Sirbu. Spv: Secure path vector routing for securing bgp. In *SIGCOMM*, 2004.

[20] 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 *FOCI*, 2011.

[21] E. Katz-Bassett, H. V. Madhyastha, V. K. Adhikari, C. Scott, J. Sherry, P. Van Wesep, T. Anderson, and A. Krishnamurthy. Reverse Traceroute. In *NSDI*, 2010.

[22] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan. Rate control for communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 49(3):237–252, 1998.

[23] S. Kent, C. Lynn, and K. Seo. Secure border gateway protocol (S-BGP). *IEEE Journal on Selected Areas in Communications*, 2000.

[24] J. Kong and X. hong. Anodr: Anonymous on demand routing with untraceable routes for mobile ad-hoc networks. In *MobiHoc*, 2003.

[25] M. E. Kounavis, X. Kang, K. Grewal, M. Eszenyi,

S. Gueron, and D. Durham. Encrypting the internet. In *SIGCOMM*, 2010.

[26] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, and F. Jahanian. Internet inter-domain traffic. In *SIGCOMM*, 2010.

[27] X. Liu, A. Li, X. Yang, and D. Wetherall. Passport: secure and adoptable source authentication. In *NSDI*, 2008.

[28] J. K. MacKie-Mason and H. Varian. Pricing congestible network resources. *Journal on selected areas in communications*, 13(7):1141–1149, 1995.

[29] H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani. iPlane: An information plane for distributed services. In *OSDI*, 2006.

[30] N. F. Magdalena, M. Balazinska, G. Harfst, H. Balakrishnan, and D. Karger. Infranet: Circumventing web censorship and surveillance. In *In Proceedings of the 11th USENIX Security Symposium*, 2002.

[31] R. Morris, E. Kohler, J. Jannotti, and M. F. Kaashoek. The Click modular router. In *SOSP*, 1999.

[32] J. Naous, M. Walsh, A. Nicolosi, D. Mazieres, M. Miller, and A. Seehra. Verifying and enforcing network paths with ICING. In *CoNext*, 2011.

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

[34] R. Perlman. Network Layer Protocols with Byzantine Robustness. MIT/LCS/TR 429, Laboratory for Computer Science, Massachusetts Institute of Technology, 1988.

[35] R. Perlman. Routing With Byzantine Robustness. Sun Microsystems Technical Report 2005-146, 2005.

[36] B. Raghavan, T. Kohno, A. C. Snoeren, and D. Wetherall. Enlisting ISPs to improve online privacy: IP address mixing by default. In *International Symposium on Privacy Enhancing Technologies*, 2009.

[37] B. Raghavan and A. C. Snoeren. A system for authenticated policy-compliant routing. *IEEE/ACM Transactions on Networking*, 2007.

[38] M. G. Reed, P. F. Syverson, and D. M. Goldschlag. Anonymous connections and onion routing. *IEEE Journal on Selected Areas in Communications*, 16(4):482–494, 1988.

[39] R. Sherwood, B. Bhattacharjee, and A. Srinivasan. P5: A protocol for scalable anonymous communication. In *IEEE Symposium on Security and Privacy*, 2002.

[40] E. Shi, I. Stoica, D. Andersen, and A. Perrig. OverDoSe: A generic DDoS protection service using an overlay network. Technical report, Carnegie Mellon University, 2006.

[41] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana. Internet indirection infrastructure. In *SIGCOMM*, 2002.

[42] Redacted for anonymous reviewing.

[43] M. Walfish, M. Vutukuru, H. Balakrishnan, D. Karger, and S. Shenker. DDoS defense by offense. In *SIGCOMM*, 2006.

[44] D. Wendlant, D. G. Andersen, and A. Perrig. Bypassing network flooding attacks using fastpass. Technical report, Carnegie Mellon University.

[45] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An integrated experimental environment for distributed systems and networks. In *OSDI*, 2002.

[46] E. Wustrow, S. Wolchok, I. Goldberg, and J. A. Halderman. Telex: Anticensorship in the Network Infrastructure. In *Proc. of the USENIX Security Symposium*, 2011.

[47] A. Yaar, A. Perrig, and D. Song. SIFF: A stateless internet flow filter to mitigate DDoS flooding attacks. In *IEEE Symposium on Security and Privacy*, 2004.

[48] X. Yang, D. Clark, and A. W. Berger. NIRA: A new inter-domain routing architecture. In *NSDI*, 2008.

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

[50] D. Zhu, M. Gritter, and D. R. Cheriton. Feedback based routing. In *SIGCOMM*, 2003.