Passport: Secure and Adoptable Source Authentication

Xin Liu Ang Li Xiaowei Yang
University of California, Irvine
{*xinl, angl, xwy*}*@uci.edu*

David Wetherall
Intel Research Seattle & University of Washington
djw@cs.washington.edu

ABSTRACT

We present the design and evaluation of Passport, a system that allows source addresses to be validated within the network. Passport uses efficient, symmetric-key cryptography to place tokens on packets that allow each autonomous system (AS) along the network path to independently verify that a source address is valid. It leverages the routing system to efficiently distribute the symmetric keys used for verification, and is incrementally deployable without upgrading hosts. We have implemented Passport with Click and XORP and evaluated the design via micro-benchmarking, experiments on the Deterlab, security analysis, and adoptability modeling. We find that Passport is plausible for gigabit links, and can mitigate reflector attacks even without separate denial-of-service defenses. Our adoptability modeling shows that Passport provides stronger security and deployment incentives than alternatives such as ingress filtering. This is because the ISPs that adopt it protect their own addresses from being spoofed at each other's networks even when the overall deployment is small.

1. INTRODUCTION

Source authentication in this paper refers to the verification of the source address of a host or network location that originates a packet. The current Internet design trusts each host to place its own IP source address on the packets that it originates, and has at best weak mechanisms to verify that a source address is correct once a packet has entered the network. Because of this, compromised hosts can place incorrect source addresses on packets to impersonate other hosts or obscure the origins of their packets, a practice known as source address spoofing.

Source address spoofing undermines the security and reliability of the Internet in a variety of ways. It enables reflector attacks, in which attackers send initial requests that spoof the address of a victim and trick hosts that reply to send unwanted traffic to the victim. Spoofing obscures the true source of the attack and amplifies it when reply packets are larger than initial requests. Reflector attacks in the early 2006 used DNS servers as unwitting participants to flood the victims with up to 5 Gbps traffic [39].

Spoofing also complicates measures to stop packet floods within the network. Packets' source addresses do not clearly identify the hosts that send them because of spoofing. Consequently, the network cannot automatically filter packet floods based on source addresses, as attackers may deliberately spoof legitimate hosts' addresses, and filtering would block the legitimate hosts' traffic as well. Similarly, fair queuing based on source addresses cannot effectively limit bandwidth consumed by packet floods. This situation creates a vicious cycle in deploying automated DoS defense mechanisms [17]: the possibility of spoofing leads to the absence of automated defense mechanisms; the absence of such mechanisms obviates the need to attack with spoofed packets, which leads back to the lack of deployment of anti-spoofing mechanisms that makes spoofing possible.

For these reasons, the Internet would benefit from stronger source authentication that makes source addresses trustworthy. Previous work that tackles this problem highlights two extremes in how it can be accomplished. One extreme is ingress filtering [16] in which each AS voluntarily filters spoofed traffic it would originate near the sources, where the legitimate source address ranges are known. This approach is light-weight, but offers limited security benefit and has incentive issues. Specifically, if one network fails to filter spoofed packets, compromised hosts in its network can spoof the addresses of other networks. Up-to-date measurements show that approximately 20% of the prefixes, IP addresses, and ASes on the Internet still allow source address spoofing [7] despite the fact that ingress filtering has been standardized as an Internet Best Current Practice for over seven years. Even when ingress filtering is deployed by all ASes, attackers may still inject spoofed packets if they could compromise routers [42, 50]. This means that ingress filtering provides no guarantee to an AS that it will not have its addresses spoofed at other parts of the network or will not become the victim of reflector attacks, even if the majority of ASes have deployed ingress filtering.

The other extreme is to use strong cryptography-based authentication to verify the source addresses. One example is the approach proposed in [33]. A packet carries a digital signature signed with a source's private key; a router verifies the signature before forwarding the packet. This approach has the adoptability benefit of allowing each AS to independently authenticate the source of a packet without relying on the deployment status or trustworthiness of other parts of the network. As long as an AS has deployed signatures, no attackers can spoof its source address at other ASes where authentication is deployed. However, it requires a per-source public key infrastructure (PKI), and routers need to verify digital signatures at line speed. Both of these requirements are steep and effectively prevent the use of digital signatures at a low level in the protocol stack.

The focus of our work is to understand whether it is possible to achieve the best of both these extremes. Ideally, we would like a source authentication scheme that is as lightweight and incrementally deployable as ingress filtering, yet as robust and beneficial in terms of incentives as digital signatures. To this end, we present the design and evaluation of Passport, a novel network-layer source authentication system. Passport treats an AS as a trust and fate-sharing unit, and authenticates the source of a packet to the granularity of the origin AS. It uses efficient symmetric-key cryptography and checks packets only at administrative boundaries. It leverages the routing system to simply and efficiently manage keys by piggybacking a Diffie-Hellman key exchange on routing advertisements. Together, these properties provide much of the benefits of digital signatures without the corresponding PKI and computational problems.

We implement a prototype of Passport and evaluate its costs and benefits via experiments, security analysis, and adoptability modeling. Our results show that a commodity software PC router can generate or verify packets at up to 2Gbps with an average packet size. We also run experiments on the Deterlab [11] testbed to show how Passport can weaken reflector attacks. We use the adoptability modeling framework presented in [10] to compare the security benefit of Passport with partial deployment with that of other approaches. Our analysis shows that Passport provides stronger security benefit with partial deployment, and hence it is more adoptable than non-cryptography-based approaches such as ingress filtering [16] and route-based filtering [26, 31].

The rest of the paper describes the design, implementation, and evaluation of Passport in greater detail.

2. PROBLEM

2.1 Source Authentication Goals

Our goal is to use source authentication to prevent spoofing under the threat model given in the next subsection. A perfect scheme would check every packet that arrives at each router and verify that the packet carries the source address of the host that injects it into the network; packets with spoofed addresses would be identified precisely and discarded. However, this perfect scheme is unattainable, and we relax the goals of source authentication to permit more realistic designs.

First, we relax the granularity of source authentication. Our design treats an AS as a trust and fate-sharing unit. That is, it only prevents hosts in one AS from spoofing the addresses of other ASes. As each AS is separately administered, we consider it to be an internal issue for an AS to prevent a malicious host in its network from spoofing the addresses of other hosts in its network. Each AS can use whatever method it prefers to do so. Note that this has the further benefit of allowing source addresses to be authenticated only at the boundaries between ASes, rather than at every router.

Second, we do not verify the freshness of each packet. That is, we do not distinguish between an authentic packet and a replay (a subsequent copy) of the same packet that is injected along the same network path as the authentic packet. This has the downside that packets may be duplicated at any point along the network path and still be considered authentic. While it would be desirable to weed out duplicates, this requires more processing than what we think belongs in the lowest layer of source authentication, e.g., per-source sequence numbers with state kept in the network. Moreover, it is not clear that duplicate packets are as problematic as spoofed packets. Small numbers of duplicates can be detected at end-systems by provisions above the network layer. Large numbers of duplicates could be filtered out in the network in cases where the benefits outweigh the costs. Or even if they are not, large numbers of duplicate packets would likely help in pinpointing the location of duplication.

2.2 Threat Model

The key threat we are concerned with is that attackers can send packets with source addresses that belong to other hosts or routers, yet have those packets pass source authentication checks in the network. We assume that in a realistic Internet environment, both hosts and routers can be compromised, although routers are compromised less often than hosts. This leads us to consider three types of attackers, each with different capabilities.

- **Compromised Host**: This attacker can inject packets into the network with arbitrary source addresses, but cannot eavesdrop the traffic sent by legitimate sources. It is referred to as a Host attacker.
- **Compromised Monitor and Host**: This attacker can eavesdrop traffic (sent by legitimate sources) at its network location and replay that traffic from other, compromised host locations in the network. It is referred to as a Monitor attacker.
- **Compromised Router and Host**: This attacker can eavesdrop traffic (sent by legitimate sources) at its location, and alter or replay that traffic at its location as well as replay it from other, compromised host locations in the network. It is referred to as a Router attacker.

Note that a Router attacker is a strong adversary that can cause greater harm than source authentication failures. We consider such attackers to better understand the

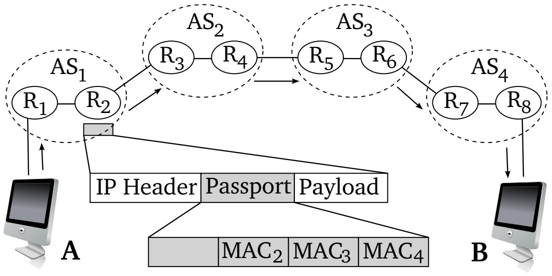


Figure 1: **A border router of a source AS (R_2) stamps source authentication information into the Passport header of an outbound packet. A border router of an intermediate or destination AS (R_3, R_5, or R_7) verifies this information.**

properties of our design. We also study the weaker Monitor attackers to tease apart the properties of Passport.

3. DESIGN

This section describes the baseline design of Passport. For clarity, we ignore the deployment issues in this section and discuss them in later sections.

3.1 Overview

Figure 1 shows how Passport works at a high level. When a packet leaves its source AS, the border router stamps one Message Authentication Code (MAC) for each AS on the path into its Passport header. Each MAC is computed using a secret key shared between the source AS and the AS on the path.

When the packet enters an AS on the path, the border router verifies the corresponding MAC value using the secret key shared with the source AS. A correct MAC can only be produced by the source AS that also knows the key. If the MAC verifies, it is sufficient to show that the packet comes from the source AS indicated by its source address. Otherwise, it is an indication that the source address is spoofed, or there is a temporary routing inconsistency. A packet with an invalid MAC is demoted at an intermediate AS and is discarded at a destination AS (§ 3.4). Routers forward demoted traffic in a separate queue with limited bandwidth (§ 4.3).

Next, we describe how two ASes obtain a shared secret key, and how MACs are computed and verified.

3.2 Obtaining Shared Secret Keys

Passport uses the inter-domain routing system for key distribution. It piggybacks a Diffie-Hellman key exchange [12] on BGP routing advertisements. Each AS_i generates a Diffie-Hellman public/private value pair (b_i, r_i), and includes the public value b_i in its routing advertisement. This routing advertisement will reach all other ASes so that they can set up a forwarding entry to reach AS_i. Similarly, AS_i will receive routing advertisements from all other ASes. Using the public values included in the routing advertisements, AS_i is able to obtain a shared secret key with every other AS_j using a standard Diffie-Hellman construct: $K(i,j) = b_j^{r_i} \ mod \ p = b_i^{r_j} \ mod \ p$, in which p is a system-wide parameter. If an AS originates multiple address prefixes, it only needs to choose one representative prefix to piggyback the key exchange information.

AS_i may receive a routing advertisement originated by AS_j from multiple neighbors. AS_i accepts AS_j's public value in the routing advertisement from its next-hop neighbor to reach AS_j. This binds the security of the Diffie-Hellman key exchange to routing security, because the public value of AS_j accepted by AS_i is forwarded from the reverse forwarding path from AS_i to AS_j. If the routing system successfully prevents AS_i from forwarding packets via an attacker by rejecting a routing advertisement forwarded from the attacker, then the public value of AS_j accepted by AS_i is not forwarded from an attacker. AS_i is able to compute a correct key shared with AS_j, and verify packets from AS_j using that key. Therefore, as long as routing is secure, Passport is secure.

Passport gains additional benefits from distributing the shared secret keys within the routing system. First, it can bootstrap the key distribution. Key distribution is a seemingly chicken-and-egg problem: keys are needed for packet forwarding, but ASes need to send packets to negotiate keys. As routing packets are exchanged before forwarding state is set up, routing has its own authentication mechanisms to accept routing messages without requiring Passport headers, i.e. routers only accept routing messages from known peers. Second, it gains efficiency. Each AS only needs to send one routing advertisement to establish a shared secret key with every other AS. Lastly, its key distribution channel can be made highly resilient to DoS flooding attacks, because the key distribution information enjoys the same forwarding priority as routing messages. If routers forward routing messages with highest priority, Passport's key distribution information is also forwarded with highest priority.

3.3 Stamping

Passport uses efficient symmetric key MACs as the inter-AS authentication information. A border router of an AS stamps a MAC for each AS on the path to the destination when it receives an outbound packet from an internal interface. Each MAC is computed using the key it shares with the AS on the path. The MAC computed for the destination AS covers the source address, the destination address, the IP identifier (IP ID), the packet length field of the IP header, and the first 8 bytes of packet payload. For instance, in Figure 1, when a packet from host A to host B leaves AS_1, the border router R_2 of AS_1 computes MAC_4 for the destination AS_4 as $MAC_{K(AS_1,AS_4)}$ (src, dst, len, $IPID$, $payload[0,7]$). The MAC computed for an intermediate AS also includes the previous AS number. For instance, R_2 computes MAC_3 as $MAC_{K(AS_1,AS_3)}$

$(src, dst, len, IPID, payload[0,7], AS_2)$. A router can obtain the AS path information from BGP.

A MAC computation covers the source address field to prevent spoofing. It covers the other fields to detect packets that are sniffed on one path but injected at other network locations. We discuss more on how Passport prevents this type of sniffing and replaying attack in § 7.

Figure 2 shows a Passport header format used in our implementation. A destination MAC is 64-bit long. Each intermediate MAC is 32-bit long if there are more than one intermediate hop to save header space; otherwise it is 64-bit long.

A border router only stamps a Passport header for a packet with a valid source address that is within its own address space, and discards the packet otherwise. This step is similar to egress filtering [16], but it is only an optimization. Passport prevents address spoofing even without it. This is because if a router stamps MACs for a source address outside its address space, the MACs will not verify at downstream ASes (as we will see next), wasting an AS's processing power.

3.4 Verification

When an AS receives a packet from an external interface, it verifies the Passport header using the key it shares with the source AS. The verifying AS uses the source address of the packet to look up the source AS, obtains the shared key, and recomputes the MAC using the shared key and the same packet fields as used by the source AS. An AS can obtain the mapping between a source address and the corresponding source AS from BGP using the AS_PATH path attribute.

If the source address is not spoofed, a router is able to locate the correct key. The re-computed MAC will match the one in the Passport header. This verifies the source AS of the packet. A router erases the MAC value in a packet after verification to prevent offline cryptanalysis.

If the MAC does not verify, a destination AS discards the packet, because the source address must be spoofed. An intermediate AS demotes the packet, because a MAC mismatch may be caused by either address spoofing or temporary routing inconsistency.

If a packet is demoted, a demotion bit in its Passport header is set, and its IP header is also marked with demotion information to convey the demotion status to legacy ASes (§ 4.3). Intermediate ASes forward demoted traffic in a separate queue with limited bandwidth without further verification. A destination AS still verifies a demoted packet, and discards the packet if the MAC is incorrect. If the MAC is valid, the packet is forwarded to its destination host with the demotion mark. End systems may use these marks to mitigate reflector attacks (§ 7.2). We discuss the trade-off between demote and discard in § 9.1.

An intermediate AS discards packets received from an external interface that spoof its own addresses. This step precedes Passport header verification, as it does not require verifying a Passport header.

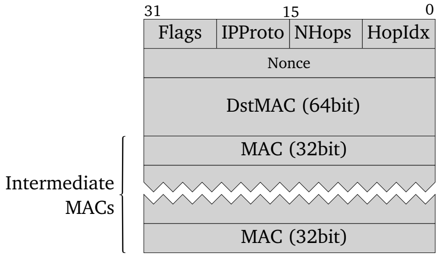


Figure 2: **Passport header format.**

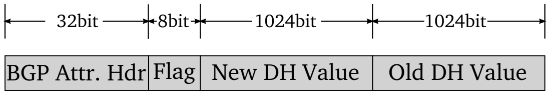


Figure 3: **Diffie-Hellman key exchange information is encapsulated in a BGP AS path attribute.**

A router in an AS may receive a packet with a Passport header from an internal interface. If the internal interface is a host-to-router interface, e.g, the interface between host A and a router R_1 in Figure 1, the router discards the packet, as a host can not generate a valid Passport header. If it is a router-to-router interface, it may assume that the packet has been verified by a border router in its AS and forward the packet without further verification.

3.5 Re-Keying

An AS may establish new shared keys with other ASes when its old keys have been used for a while, e.g. on the order of a few days or a few weeks, to improve security. To exchange new keys, AS_i sends a routing update with a new Diffie-Hellman public value, and other ASes compute new keys shared with the AS using this value.

During the re-keying process, different ASes may use different keys to generate the MACs for AS_i, as the routing advertisement of AS_i will arrive at different ASes asynchronously. To identify different keys, an AS uses an alternating parity bit to mark consecutive Diffie-Hellman public values. When AS_j generates a MAC for AS_i, it uses the highest-bit in the MAC field to indicate the parity of AS_i's public value, and one bit in the Flags field of a Passport header to indicate the parity of its own public value. These two bits uniquely identify a shared key even when both ASes re-key simultaneously.

Figure 3 shows the key exchange information piggybacked in a BGP advertisement. Each advertisement carries both the new and old Diffie-Hellman public values in case a remote router crashes when an AS re-keys. The parity of the new value is indicated in the flag field.

3.6 Key Management and Storage

Diffie-Hellman public values or the shared secret keys are stored with routing information. If a router reboots and loses those values, it may obtain them from other routers in the same AS, similar to a BGP table transfer after a router reboots.

An AS may configure one router or a route reflector [5] as its key generator. This router is in charge of generating Diffie-Hellman value pairs and initiating re-keying. Other routers learn the Diffie-Hellman private value from the key generator via iBGP.

4. DEPLOYMENT

This section describes how to deploy Passport in the presence of various legacy elements in the Internet. We discuss the high-level issues here. More details such as MTU discovery are discussed in § 9.

4.1 Inter-Operate with Legacy ASes

ASes that adopt Passport may use the optional and transitive path attributes of BGP to distribute their Diffie-Hellman public values as shown in Figure 3. Legacy ASes do not process the optional and transitive path attributes, but will include them in the routing advertisements they propagate to their neighbors [36]. Therefore, two upgraded ASes can perform a Diffie-Hellman key exchange when there are legacy ASes between them.

Passport header is inserted as a shim layer between IP and an upper layer protocol. A source AS stamps MAC values for the upgraded ASes on the path, and legacy ASes do not process the Passport header.

4.2 Bump in the Wire

Passport can be deployed as *bump in the wire* without upgrading hosts. When both a source AS and a destination AS have upgraded, the border router at the source AS inserts a Passport header to a packet, and the border router at a destination AS strips off the header.

If a destination AS has not deployed Passport , an upgraded source AS can still use Passport headers so that other upgraded ASes on the path can verify its packets. In this case, the last upgraded AS on the path strips off the Passport header. However, if there is a temporary routing inconsistency, a Passport header may not be stripped off when the packet reaches its destination. A legacy host in the destination AS may receive a packet with an unknown Passport header and discard it.

To solve this problem, Passport uses IP encapsulation. A source AS encapsulates the original IP packet using an outer IP header. The outer IP header uses the same source address as the inner header, and uses the last upgraded AS on the path as the destination address. This address could be a special well-known anycast address of the last upgraded AS. The source AS inserts a Passport header between this outer IP header and the inner IP header. In this encapsulation mode, a MAC in a Passport header covers the source address, both destination addresses in the inner and outer header, the original 8-byte payload, and the IP ID and length fields of the outer header.

When the destination AS in the outer IP header decapsulates a packet, it checks whether the incoming AS is a neighbor to which it announces the destination prefix in the inner header. If not, it discards the packet to prevent a source AS from violating its transit policy.

Modern routers already support encapsulation at line speed because of other needs such as VPNs and IPv4-IPv6 transition. After hosts are upgraded, they can process packets with Passport headers, reducing the need for encapsulation.

4.3 Handling Legacy Traffic

In upgraded ASes, legacy and demoted traffic are both unverified traffic, and are treated with the same priority. An upgraded AS may use strict priority queuing to favor verified traffic over unverified traffic, and incent legacy ASes to upgrade. Alternatively, it may use two weighted queues to handle verified and unverified traffic, allocating limited bandwidth to unverified traffic. It may set the queue weights according to the ratio of traffic from upgraded ASes and legacy ones to avoid penalizing legacy traffic when there is no spoofing attack.

An upgraded AS will discard or demote legacy traffic if it detects or suspects that the traffic spoofs other upgraded ASes' addresses as follows:

1. If a legacy packet's source AS and destination AS have both deployed Passport, discards the packet, as it must be spoofed.

2. If a legacy packet's source AS has deployed Passport but the destination AS has not, demotes the packet. This is because a source AS will insert a Passport header for a legacy destination if there is any upgraded AS on the path. A legacy packet from an upgraded AS is likely to be spoofed, except during temporary routing inconsistency.

A legacy AS may receive two types of legacy traffic: regular and demoted by other upgraded ASes. The AS should treat demoted traffic with lower priority, because it is likely to be spoofed.

Upgraded ASes convey demotion information to legacy ASes via the IP header, such as using the Differentiated Services Code Point (DSCP) [29]. A legacy AS needs to make a configuration change to honor demotion in order to take advantage of Passport. We believe this configuration change can be made at most legacy ASes, because DiffServ is already well supported by commercial routers today, and this change does not require software or hardware upgrade. Besides, if a legacy AS encounters congestion in its network, it has to discard some packets. It's advantageous for the AS to honor a demotion mark and discard packets that are likely to be spoofed in favor of regular packets.

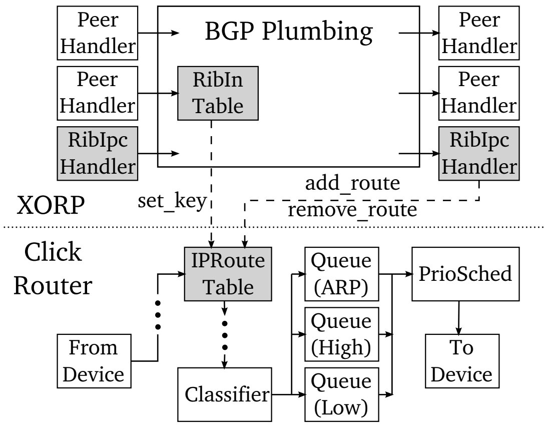


Figure 4: **How Passport is implemented using Click and XORP. The shaded boxes are the main modules we modify.**

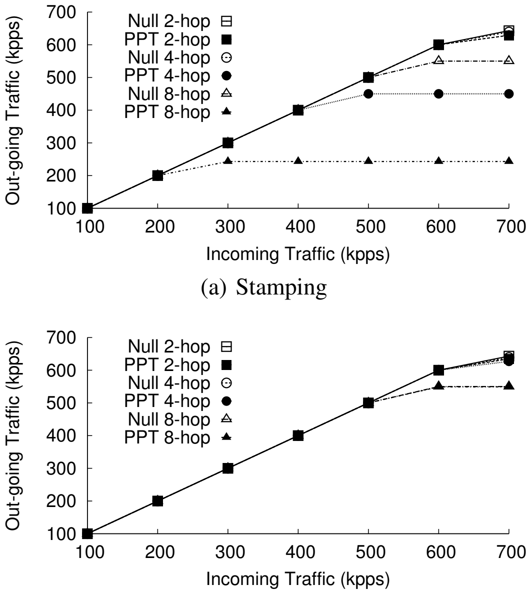
5. IMPLEMENTATION

This section describes a prototype implementation of Passport. We implement most of the features of Passport using Click [22] and XORP [18]. Figure 4 shows the structure of the implementation. The shaded boxes are modules we modify. We modify XORP to piggyback the Diffie-Hellman key exchange protocol in BGP, and modify components in Click to support Passport header stamping and verification. We also modify XORP to communicate with Click using the `/click` file system.

We add an optional and transitive AS path attribute DH_KEY to XORP's BGP modules. This attribute encapsulates Diffie-Hellman public values as described in Figure 3. It is inserted into the BGP prefix advertisements in the module `RibIpcHandler` and extracted from the prefix advertisements in the module `RibInTable`. `RibInTable` is also modified such that whenever new Diffie-Hellman public values are received, the corresponding shared secret key is generated and sent to Click using the `set_key` interface in Figure 4. We also modify `RibIpcHandler` to update Click's routing table with Passport related information using the `add_route` and `remove_route` interfaces in Figure 4. Passport related information includes AS paths and prefix to origin AS mapping. In our current implementation, a new Diffie-Hellman public-private value pair is generated at XORP's startup time. This part needs to be extended to support periodic re-keying, and private key distribution using iBGP.

We modify the `IPRouteTable` element in Click such that it receives shared secret keys from XORP and calls `generate_ppt()` or `verify_ppt()` in its `push()` method to stamp or verify Passport headers. We use Click's priority scheduler to handle normal and demoted traffic as shown in Figure 4. We use priority queuing instead of weighted fair queuing to emphasize the benefit of deploying Passport . The `ARP` queue ensures that link-local ARP packets have highest forwarding priority.

Our implementation uses UMAC because of its superior speed. UMAC takes a nonce as input. We stamp a random number into the 32-bit nonce field of a Passport header and use it together with the 16-bit IP ID to generate a 48-bit nonce for UMAC computation. In the worst case when IP ID field never changes, the nonce space is 2^{32}, and a nonce may be reused before an AS rekeys. For this reason, we did not use UMAC in [23] because it is vulnerable to a single nonce reuse. Instead, we use the construction in [8] combined with UHASH [23] and AES. This construct is provably robust to occasional nonce reuse. (Due to the lack of space, we cannot include the proof, but we have confirmed our proof with the UMAC inventor.)



(a) Stamping

(b) Verification

Figure 5: **The throughput of Passport header stamping and verification for various AS hops with minimum sized packets (40 byte TCP/IP headers plus a Passport header). The Click null forwarding throughput for packets with the same size are also shown for comparison.**

6. PERFORMANCE EVALUATION

6.1 Passport Header Processing Overhead

We use three PCs in our laboratory to measure the computational overhead of Passport header stamping and verification. One PC is used as a router, connecting a packet generator PC and a sink PC. The router has an AMD Opteron 285 Dual Core 2.6GHz CPU, 2GB memory, and an Intel PRO/1000 GT Quad Port Server Adapter. The packet generator and sink are equipped with Pentium-D 3.4GHz CPU, 2GB memory and Intel PRO/1000 PT Server Adapter. We measure both the throughput and CPU cycles of Passport stamping and verification.

To measure the throughput of Passport header stamp-

	Operation	Time		
		2-hop	4-hop	8-hop
Per Packet	Passport Stamping	655 ns	1493 ns	3190 ns
	Passport Verification	578 ns	618 ns	631 ns
Re-key	DH value pair (1024-bit)	5.64 ms		
	Symmetric key (128-bit)	5.64 ms		

Figure 6: **Micro-benchmark of various Passport operations. Time is converted from CPU cycles.**

	Security	Sig. Size	Signing	Verification
RSA-512	60-bit	64 bytes	512 us	40 us
RSA-1024	72-bit	128 bytes	2214 us	102 us
DSA-512	65-bit	40 bytes	368 us	443 us
ECDSA-160	78-bit	40 bytes	300 us	1400 us

Figure 7: **Equivalent MAC security level, signature size and computational costs of well-known public key schemes.**

ing, we let the packet generator send minimum sized packets (40 bytes TCP/IP headers) at various input rates. Our experiments assume legacy hosts, and the router PC inserts Passport headers into the packets. The sink measures the output rate. We vary the number of AS hops for each experiment, because a router needs to stamp a MAC for each AS on the path. Note that N AS hops implies $N - 1$ MAC computations. Ten million packets are sent for each experiment.

Similarly, to measure the throughput of Passport header verification, we let the packet generator send minimum sized packets with preset Passport headers at various rates with various AS hops, use the router PC to verify Passport headers, and measure the output rates at the sink.

Figure 5 shows the Passport header stamping and verification throughput, together with Click null forwarding throughput for packets of the same size. The Passport header verification throughput matches well with Click's null forwarding throughput, as it only involves one MAC computation. The verification throughput varies from 636 kpps to 549 kpps for Passport headers of two to eight AS hops. The slight decrease is primarily due to the increase of packet length, not by the MAC computation.

The Passport header stamping throughput decreases as the AS path length increases. For Passport headers with two to eight AS hops, the throughput varies from 628 kpps to 243 kpps. If we assume that the average packet size is 400 bytes [1], the PC router can stamp Passport headers for average sized packets with two to eight AS hops at 2 Gbps to 0.9 Gbps. As the mode and mean of the AS path length are between 3 and 4 [13], we expect that the stamping throughput for real Internet traffic exceeds 0.9 Gbps. We also note that an AS only needs to stamp Passport headers for traffic originated within its network, and not for transit traffic. 1∼2 Gbps stamping throughput might be sufficient for most ASes.

We measure the CPU cycles for Passport header stamping and verification using the *get_cycles()* Linux kernel function. Figure 6 shows the result, with CPU cycles converted to time. The per-hop increment of a Passport header stamping time is about 420 ns.

Passport-enabled routers also exchange Diffie-Hellman keys on the routing plane. The cryptographic operations include generating Diffie-Hellman public-private value pairs and computing shared secret keys from Diffie-Hellman values. Both Diffie-Hellman value pairs and shared secret keys are generated using exponentiation. We tested the overhead to generate a Diffie-Hellman value pair and to compute a shared secret key on the router PC. As shown in Figure 6, each public key operation takes 5.64 ms.

The public key operations are expensive, but are unlikely to become a performance bottleneck, because an AS only performs public key operations when it re-keys. Re-keying should happen infrequently such as no more than once per week. When an AS itself re-keys, it needs to generate a shared secret key with all other ASes. There are less than 30K ASes seen in BGP routing tables according to data from RouteViews [37]. It takes less than three minutes to generate all shared keys on the router PC. If another AS re-keys, an AS only needs to generate one shared secret key when it receives a new Diffie-Hellman public value from that AS. If we assume all ASes re-key randomly during a period of seven days, then on average, an AS may receive less than three new Diffie-Hellman public values per minute, and spend 17 ms to compute the shared secret keys.

6.2 Memory Overhead

Passport maintains per-AS key information. A router keeps a shared secret key per AS for Passport header stamping and verification. A MAC computation typically requires the initialization of a key context. It is desirable that a router initializes and stores the key context for fast processing. The size of a key context depends on the specific MAC. Our implementation uses a UMAC key context that consumes 388 bytes. It requires less than 12MB memory to store the shared keys and their key contexts.

When an AS re-keys, another AS may need two different keys for Passport stamping and verification: one generated with the old Diffie-Hellman public value, and the other with the new value. This requires additional memory. As shown above, the average re-keying rate of all ASes is less than 3 per minute. If we assume it takes at most an hour for BGP to converge, then the number of simultaneously re-keying ASes is around 180, adding an additional 70KB memory overhead.

6.3 Header Overhead

As shown in Figure 2 and described in § 3.3, a Passport header has a 16-byte fixed header overhead, and four bytes per additional AS hop if the path length exceeds 4 hops, or an additional 8-byte overhead if the path length is 3 or 4. If we assume an average AS path length is four,

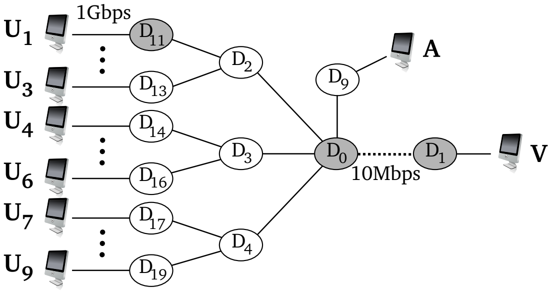


Figure 8: **Topology of Deterlab experiments.**

the average header overhead is $8 + 8 + 8 = 24$ bytes.

Passport's header and processing overhead is inherent to cryptography-based security mechanisms. For comparison, we list the header and processing overhead of well-known public key signatures that provide a similar level of security as one 64-bit MAC in Figure 7. The tests are done using the *openssl* speed test on the router PC, and the security levels are estimated according to [25].

6.4 Deterlab Experiments

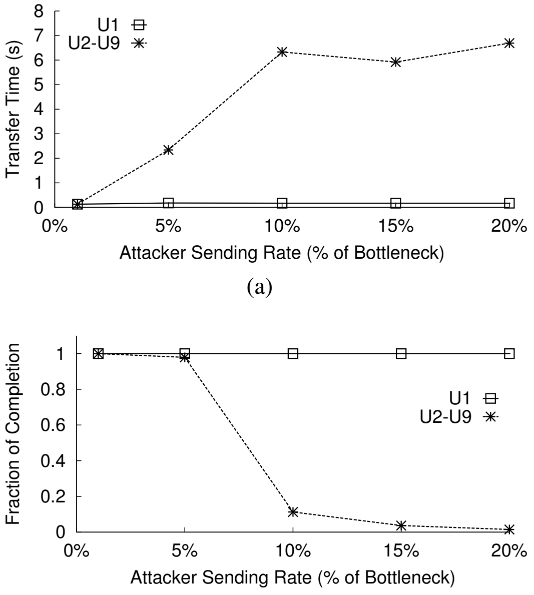
We run experiments on a Deterlab [11] testbed to test the correctness of our implementation. We emulate a scenario in which malicious Host attackers in legacy ASes spoof the source address of a victim in an upgraded AS to launch reflector attacks.

The experiment topology is shown in Figure 8. Each circle represents one AS. Only the shaded ASes deploy Passport. We configure each AS to have only one router D_i. The bottleneck link is between D_0 and D_1. The victim host V is connected to D_1, the attacker host A is connected to D_9, and D_{11} to D_{19} each has one host connected to it. We wish to use a topology with more hosts, but we were limited by the number of PCs we could hold on the Deterlab.

In our experiments, hosts U_1 to U_9 each run a reflector application that replies to incoming UDP packets with an amplification ratio of 40. The attacker spoofs the victim's address and sends UDP packets to all reflectors $U_1 \sim U_9$ uniformly. Each UDP packet sent by the attacker has 32 bytes payload. These parameters are set to emulate a DNS reflector attack [39].

After the attack is started, hosts U_1 to U_9 also each send 100 files to the victim using TCP. These TCP transfers are used to measure how the reflector attack affects the network performance seen by an end system. The size of each file is 20KB, and a file transfer aborts if it cannot finish in 10 seconds. We vary the attacker's sending rate from 1% to 20% of the bottleneck link bandwidth and measure the file transfer time.

The results are shown in Figure 9. Hosts U_2 to U_9 are in non-upgraded ASes. Their file transfer traffic is legacy traffic and competes for bandwidth with the legacy reflector traffic at D_0. When the reflector traffic congests the bottleneck link, their file transfer traffic suffers from congestion. The file transfer traffic from U_1 carries Passport headers and only competes for bandwidth with verified Passport traffic at D_0. Therefore, U_1 is not affected by the reflector traffic and can finish all the file transfers quickly. Note that the reflector application on U_1 will not receive attack traffic and therefore will not send out reflector traffic to compete with U_1's file transfer traffic at the bottleneck link, because when attack packets to U_1 reach D_0, D_0 will discard them as they do not include Passport headers but both their source and destination addresses belong to upgraded ASes (See § 4.3).



(a)

(b)

Figure 9: **The average file transfer times and fractions of completion for hosts U_1 to U_9 when the attacker A spoofs V's address to launch a reflector attack. U_1 is in an upgraded AS D_{11}, while U_2 to U_9 are in non-upgraded ASes D_{12} to D_{19}.**

7. SECURITY

7.1 Spoofing Prevention

Host Attacker: To spoof source addresses, a Host attacker may try to break the MAC that is the heart of Passport. But our design uses a standard MAC scheme with 128-bit keys, which is computationally infeasible to break. The attacker might instead try to guess a valid Passport header by sending packets. Since a Passport header has at least a 63-bit destination MAC value (modulo the parity bit of a Diffie-Hellman public value), an attacker would expect to send at least 2^{62} packets to guess one correct, but the time it takes to send those packets exceeds the period for which the Passport header will be valid. The long-term key distributed via BGP will have advanced in the interim. This attack would also signal a clear anomaly via a large number of invalid Passport headers.

An attacker may try to guess an intermediate 31-bit MAC value by sending TTL expired packets, and use the echoed IP header and Passport header to observe whether a guess is demoted. But this again is infeasible as ICMP messages are always rate limited.

Monitor Attacker: An eavesdropper may observe valid Passport headers but cannot freely transfer them to another path because a Passport header is bound to one AS path, i.e., its MACs will be found invalid if sent via another AS path. Thus, packets transferred to other paths will be demoted on their paths to destinations, and can not compete for bandwidth with packets with valid MACs.

Router Attacker: Packets duplicated by routers on the forwarding path of a source may reach a destination without being demoted. However, in this case, routing is compromised, and Passport's security is bound to routing security. An AS spoofed by a router on its forwarding path should choose a different path.

If an attacker compromises an AS, it may use the AS's keys at other, unprotected locations in the network to forge a Passport header that spoofs the AS's addresses. But this only implicates the compromised AS, not other parts of the network. Similarly, a compromised router can spoof packets from other addresses in its AS, but this again implicates the AS and may adversely affect its traffic.

Importantly, even if an AS is compromised, it cannot forge a Passport header from another AS. This is because it does not have the secret keys of the other AS. Even colluding ASes can only forge a valid Passport header that shows a packet comes from themselves. They cannot forge a Passport header as if the packet were from an AS outside the colluding set.

We also note that although two ASes share a secret key, they can not use this key to spoof each other's addresses at other ASes, because other ASes use separate keys shared with these two ASes respectively to validate their addresses.

The security of Passport is bound to routing security. We rely on routing to distribute the correct public Diffie-Hellman public values across ASes. By the Diffie-Hellman construction, we do not rely on routers to keep the public values secret from attackers, since it is computationally infeasible to find the long-term pair-wise keys given only the public values. However, if an attacker can successfully hijack a prefix announcement and replace the Diffie-Hellman public value, it can both receive packets for the specific prefix, and send packets as if they were originated from that prefix.

7.2 Reflector Mitigation

A Monitor Attacker may attempt to launch reflector attacks by sniffing packets sent by a victim, and injecting duplicate copies of them from other network locations. Those replayed packets will be demoted because Passport is bound to an AS path. An upgraded host that understands the demotion mark in an IP header should echo back the demotion mark in its reply packets to avoid becoming a reflector.

Unfortunately, if a host is not upgraded to echo back a demotion mark, it may respond to replayed packets without demoting the reply packets, thereby becoming a reflector. Passport alleviates this problem by including the destination address, IP ID, IP length field of an IP header, and 8 bytes of payload in the MAC computation. The 8-byte payload covers the TCP sequence number and UDP checksum. A replayed packet must have the same source address, destination address, IP ID, IP length, and TCP sequence number or UDP checksum as the sniffed packet. These fields can help end systems detect and discard replayed packets. For instance, an upper-layer application or a transport protocol such as TCP may detect a duplicate packet, or a sequence number or checksum mismatch, thereby discarding the packet.

We believe that in practice, these mechanisms can effectively mitigate reflector attacks launched using replayed packets even if a host is not upgraded to echo back a demotion mark. However, if this type of attack becomes a serious concern, Passport can be extended to detect and discard packets with duplicate Passport headers in the network at a higher cost, using a combination of sequence numbers and bloom filters as described in an early version of this work [27].

In the case of a Router Attacker, packets can be duplicated without being demoted. As described above, an AS should avoid forwarding packets via a Router Attacker.

7.3 Security with Partial Deployment

In the incremental deployment phase, Passport prevents the addresses of upgraded ASes from being spoofed at other upgraded destination ASes by Host attackers. If a destination AS is not upgraded, then as long as there is an upgraded AS on the path, packets that spoof the upgraded ASes' addresses will be demoted.

Similarly, a packet replayed on one path but sniffed on another path by a Monitor or Router attacker will be demoted as long as the replayed path differs from the sniffed path by one link whose end node is an upgraded AS.

8. MODELING ADOPTABILITY

This section uses modeling and simulation to study the adoptability and the incremental deployment benefit of Passport. We are interested in this study because ingress filtering, despite being lightweight, offers little incentive for adoption. An AS that deploys it can still have its addresses spoofed at other parts of the network.

To examine whether Passport provides greater security benefit to motivate adoption, we use the framework presented in [10] to compare the adoptability of Passport with that of ingress filtering and SAVE [26], a protocol to establish route-based filters [31]. We compare with SAVE because to the best of our knowledge, route-based filtering is the most effective non-cryptographic method that mitigates spoofing with partial deployment [3, 31], and SAVE is the only proposal that implements accurate route-based filters.

8.1 The Model

Our adoptability model simulates the adoption decision of each AS via iterations. At each iteration, an AS S that has not deployed a spoof prevention mechanism compares its security benefit before and after adoption. If the benefit exceeds its cost threshold, S adopts the mechanism. The iterations stop when no more ASes adopt the mechanism. The critical threshold—the largest cost threshold that leads to full deployment—measures the security benefit a spoof prevention mechanism provides to an adopter. The higher the metric, the greater the benefit, as an AS is willing to pay a higher cost to adopt it.

For an AS S that considers to adopt a spoof prevention mechanism, we define the security benefit as the average probability that an attacker cannot spoof S's addresses in the network. To calculate this probability, an AS S iterates through every other AS D, and examines whether a malicious attacker at an AS M can spoof its addresses at D. A security indicator $E(M, D) \in \{0, 1\}$ is set to 1 if M can not spoof S, and 0 otherwise. The probability that an attacker cannot spoof S at D is computed as $\sum_M E(M, D)P(M)$, in which $P(M)$ is the probability that M is malicious. The security benefit of S, denoted by F is the weighted average of $\sum_M E(M, D)P(M)$ among all Ds. That is,

$$F = \frac{\sum_D \omega_D \sum_M E(M, D)P(M)}{\sum_D \omega_D} \quad (1)$$

The weight ω_D models that an AS S sends different amount of traffic to different AS Ds. Intuitively, the more traffic S sends to D, the more important that S's addresses are not spoofed at D. The weight $P(M)$ models different security levels of different ASes.

An AS S computes the security benefit F' after its adoption, and its current security benefit F without its adoption, and uses the difference $\Delta F = F' - F$ as its incentive for adoption. At each iteration, S compares ΔF with a cost metric c. If $\Delta F > c$, it adopts the spoof prevention mechanism.

In our model, we use a uniform cost metric c for all ASes to normalize the security benefit F. Larger ASes may have higher deployment costs, but they also have larger address spaces. F can be considered as benefit per address, as it does not include the size of S.

8.2 Mechanisms

We briefly introduce ingress filtering and SAVE, and describe how to compute the security benefit F for them and for Passport.

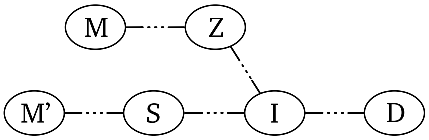


Figure 10: A sample topology.

Ingress filtering: An AS S that deploys ingress filtering can filter spoofed traffic originated from hosts in its network or from its customer ASes as well as incoming traffic that spoofs its own addresses [16]. Before an AS S deploys ingress filtering, any malicious node can spoof its address space at an AS D. After S adopts ingress filtering, an attacker can still spoof S at D, unless the attacker's traffic to D is forwarded via S, or $D = S$. For instance, in Figure 10, if M' is the malicious node, after S deploys ingress filtering, M' cannot spoof S at D.

SAVE: SAVE [26] is a proposal that automatically installs route-based filters. A router maintains an incoming table that maps a source address prefix to an incoming interface at the router. A source AS that deploys SAVE periodically sends a source address update for every destination prefix in its routing table. A router uses the incoming interface of an update message to update its incoming table. When a router receives a packet, it discards the packet if its incoming interface does not match the one associated with its source address in the incoming table.

Before an AS S deploys SAVE, any malicious node can spoof S at an AS D. After an AS S deploys SAVE, a malicious node can not spoof S at D, if there is at least one upgraded AS on the path from the AS S to D, and the incoming interface of S at the upgraded AS is different from that of the malicious node. In Figure 10, if M is an attacker and Z has deployed SAVE, then M cannot spoof S at D, because S's incoming interface to reach Z is $I \rightarrow Z$, while M's incoming interface at Z is $M \rightarrow Z$.

Passport: If an AS S does not deploy Passport, any malicious node can spoof S's addresses at an AS D. If an AS S deploys Passport, its security benefit depends on the attacker model. If a malicious node is a Host attacker, the malicious node cannot spoof S at D if there is at least one upgraded AS on the path from the malicious node to D. This is because the malicious node cannot spoof a valid Passport header.

If the malicious node is a Monitor attacker, it can sniff S's traffic sent via itself, and collude with other compromised Host attackers to replay sniffed traffic. The replayed traffic by a compromised Host attacker will be demoted if the path from the Host attacker to D and the path from S to D differ by at least one link whose end node is an upgraded AS. For instance, in Figure 10, if M is a colluding compromised Host attacker, as long as the path from M to I (including I) has one AS that deploys Passport, M can not spoof S at D without being demoted, regardless of the Monitor attacker's location. This is because an intermediate MAC in a Passport header covers the previous incoming AS number.

The security benefit under Router attacker threat is similar to that under Monitor attacker, except that a Router attacker can replay sniffed packets at its own location.

Both ingress filtering and SAVE have the same security benefit under a Monitor threat and a Host threat, because they do not insert secrets in packet headers. Under the Router attacker threat, a Router attacker on an AS S's forwarding path can always spoof S's addresses, before or after S adopts ingress filtering or SAVE.

We note that in computing the security benefit for Passport, as long as M's spoofed traffic is demoted at D, S considers its traffic not spoofable by M at D, as M's spoofed traffic is distinguishable from its authentic traffic.

8.3 Adoptability

We run simulations to compare the critical thresholds of various mechanisms. Our simulations use a generated topology of 1000 ASes, as the computational overhead is over $O(n^3)$, and we cannot finish one run on larger topologies in less than a day. For the purpose of cross validation, we use the same topology as the one used in [10]. The topology is generated using Inet [44] and BGP routing tables. We also generate smaller topologies using the same method, and run simulations on smaller topologies to confirm that the trends are consistent.

Our simulations vary the distributions of ω_D and $P(M)$, using similar models as in [10]. Due to space limitation, we only present results assuming a uniform distribution of ω_D and $P(M)$. Results using other distributions vary in the absolute values, but the trends are similar. We use 10 random initial adopters (1% of all ASes) for each run.

For the Host attacker, we simulate two different attacker populations. One assumes that a Host attacker can only be in an AS that does not deploy a spoof prevention mechanism, and the other assumes that it can be in any AS. We only show results for the first case, as the results are similar, and in the second case, Passport has stronger security benefit.

We simulate all three threat models: Host, Monitor, and Router. For Monitor and Router attackers, we assume that at most 3% of transit ASes can be compromised, and randomly pick 3% of them to be the Monitor or Router attackers. We average the security benefit over 100 runs. We assume that Host or Monitor attackers' colluding hosts can be any Host attacker.

Figure 11 shows the results. We omit the results for Router attackers for all mechanisms for clarity. Those results are very similar to the results for Monitor attackers, because in the Monitor attacker case, we already consider that a colluding Host may be on the path from S to D. As can be seen, ingress filtering has the lowest adoptability critical threshold. This is expected, because ingress filtering provides little immediate benefit to an adopter. Attackers can spoof an adopter's addresses at other ASes, before and after an AS's adoption.

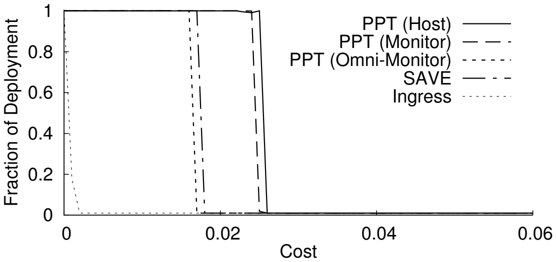


Figure 11: **The steep cliff of each curve shows the critical threshold of a spoof prevention mechanism. Ingress filtering (Ingress) has the lowest adoptability. Passport (PPT) is more adoptable than SAVE under both Host and Monitor threats. Its adoptability is still close to SAVE even if we assume an omnipresent Monitor attacker.**

Both Passport and SAVE are much more adoptable than ingress filtering. Passport has a higher adoptability threshold than SAVE. The adoptability threshold of Passport is only slightly affected by the presence of Monitor attackers, because 3% of Monitor attackers can only sniff a fraction of all paths. We also tested an unrealistic omnipresent Monitor model (Omni-Monitor in the figure), in which a Monitor attacker can sniff all S-D pairs. In this extreme case, the adoptability threshold of Passport is still very close to that of SAVE. This is because the algorithm of detecting replayed traffic under Passport is very similar to that of SAVE as described above.

8.4 Strong Security Benefit

Passport provides a strong security assurance that as long as an AS deploys it, other attackers cannot spoof its addresses at other ASes that also deploy it. It does not depend on transitive trusts between ASes to provide this assurance. However, the adoptability model does not capture this feature, because it computes the average probability of not being spoofed.

We define a strong security metric that measures the fraction of ASes at which no attackers can spoof an AS S's addresses. Referring to Equation 1, when computing a strong security metric, we only include an AS D in the sum $\sum_D \omega_D \sum_M E(M,D)P(M)$, if and only if $\sum_M E(M,D)P(M) = 1$.

Figure 12 shows the strong security metric of various mechanisms under various threats. The metric is averaged over all ASes in the network. With Passport, the fraction of ASes at which no attackers can spoof a source AS's addresses scales linearly with the number of upgraded ASes. When there are Monitor or Router attackers, they may replay the Passport headers of the sources for which they provide transit service, but do not affect the security assurances of other upgraded ASes. With Router attackers, the security metric can not reach 100%

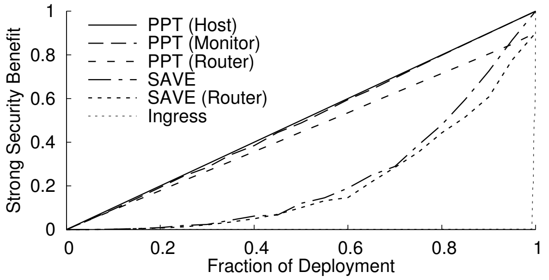


Figure 12: **The fraction of ASes at which no attackers can spoof a source AS's addresses with various fractions of deployment.**

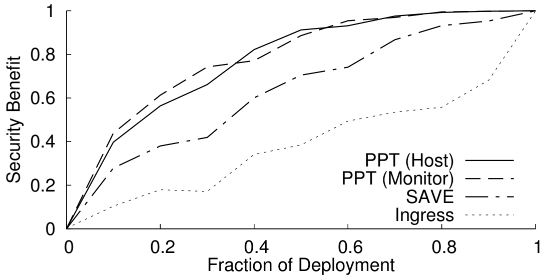


Figure 13: **The security benefit F averaged over all ASes in the network with various fractions of deployment.**

because on-path replayed packets can not be detected.

SAVE can not achieve this level of security assurance before it reaches a full deployment. This is because its security assurance depends on transit trust: if there is any non-upgraded AS on the path from a source to a destination, compromised hosts in that AS or connecting to that AS from a non-upgraded path can spoof the source AS's addresses.

Ingress filtering's security assurance is close to zero before it reaches full deployment, because any attacker can spoof the address space of an AS S, unless the attacker's provider deploys ingress filtering, or S provides transit service for the attacker. Our topology does not have customer-provider ingress filtering information. When computing the security benefit, we assume a leaf node's next hop AS filters spoofed traffic from the leaf node.

For comparison, Figure 13 shows the security metric F averaged over all ASes in the network using Eq 1. Again, for clarity we omit the results under Router attackers for Passport as they are similar to the ones under Monitor attackers. As can be seen, SAVE has a much higher security metric than the results in Figure 12. This suggests that the main security benefit of SAVE comes from incrementally reducing the probability of spoofing, while a large portion of Passport's security benefit is to prevent spoofing with certainty. The curves are not smooth because of the randomness in initial deployment.

Interestingly, Figure 13 shows that ingress filtering also has security benefit with partial deployment. However, this benefit does not motivate adoption, because it is almost the same for an AS before or after its adoption.

9. DISCUSSION

9.1 Demotion vs Discard

Passport demotes rather than discards packets with invalid MACs at intermediate ASes. Another design choice is to discard those packets. Discard has the advantage that packets will not further consume the network's resources, and legacy ASes do not need to make configuration changes. The drawback is to introduce unnecessary packet loss during routing convergence.

We choose demote over discard because a small loss rate may adversely affect TCP's performance, and presently BGP converges slowly. This design choice may not be optimal if these conditions change.

9.2 Per-Prefix Key versus Per-AS key

Our design uses per-AS key rather than per-prefix key for scalability. The memory overhead of per-prefix key is the number of prefixes one AS announces multiplying the total number of prefixes announced in BGP, a number much larger than that of per-AS key. The time it takes to recompute all shared keys after an AS re-keys is also longer. Per-AS key complicates issues such as multi-origin prefixes (as we'll describe shortly). If in the future, a router's CPU or fast memory is not a limited resource, the design can be changed to use per-prefix key.

9.3 Additional Deployment Issues

MTU Discovery: When Passport is deployed in the *bump in the wire* mode, a legacy host may not subtract the Passport header in its MTU discovery process. One solution is for a border router to intercept the ICMP "Fragmentation Needed" message, and subtract the Passport header, a practice used in the deployment of IPv4 to IPv6 [30]

Packet Fragmentation: Packets fragmented in the middle of the network will not have a valid Passport header. Passport demotes all fragments, including at the destination AS. We are not concerned much with this issue, because fragmentation by the network is discouraged, and has been disabled by IPv6.

Prefix aggregation: If an AS's prefix is aggregated by its provider AS, then its AS path attributes, including its Diffie-Hellman value will be lost. In this case, an AS should rely on its provider to stamp and verify its traffic. A customer AS that desires to stamp and verify Passport headers on its own should request its providers not to aggregate its prefix. As an AS only needs one prefix to distribute the key exchange information, even if that prefix is not aggregated, it will not significantly increase the BGP table size, as there are much more prefixes than ASes in the Internet (>244K versus <27K).

Multi-origin prefixes: BGP advertisements may have multi-origin AS conflicts (MOAS), a practice not recommended by IETF [19]. MOAS interferes with the key lookup process, as a router needs to use the correct key from the origin AS to verify a Passport header. MOAS can be a signal of prefix hijacking. In this case, Passport relies on the routing system to resolve MOAS conflicts.

MOAS can be caused by sibling ASes announcing each other's prefixes [28]. In this case, they should share Diffie-Hellman values so that a verifier can use the key shared with either AS to verify their addresses.

MOAS can also be caused by multi-homed ASes connecting to its providers without BGP [51]. In this case, a simple solution is for the site to run BGP, in order to be compatible with Passport and IETF's recommendation.

Our observation from a BGP table obtained from the Oregon RouteViews server on Aug 1st, 2007 shows that 1385 out of 244095 (0.57%) prefixes have more than one origin. Therefore, we expect that MOAS prefixes are uncommon, and are unlikely to be a deployment hurdle.

Anycast addresses have legitimate multi-origins. Passport treats packets with anycast source addresses as legacy traffic. A source should not send traffic with anycast source address. An anycast address can be used as a destination address, because when computing a MAC for a destination AS, a source AS uses the key shared with the anycast address' destination AS, as described in § 3.2.

Path Discrepancy: Mao et al. [28] observe that paths inferred by traceroute may be different from BGP paths when routing is stable. They postulate several causes. Other than prefix aggregation and MOAS, most of them are due to traceroute not accurately reflecting forwarding paths or AS boundaries. One rare cause is an iBGP misconfiguration of a transit AS. Passport can become a diagnosis tool to such routing anomalies. If a router discovers that all traffic it forwards cannot be verified, it is a strong indication of misconfiguration.

Inter-domain multicast. Passport only provides source authentication for unicast traffic, because the origin of a duplicated multicast packet does not match its source address. Routers should use separate authentication mechanisms such as [34] to authenticate multicast traffic.

Enable Passport on High Speed Routers. The performance of our prototype implementation is insufficient for high speed routers. However, with optimized hardware implementation, throughput of tens of gigabits per second may be achieved. The bottleneck of Passport header processing is the AES-based MAC computation. There are already commercially available hardware AES implementation that can encrypt at the speed of 40Gbps [20]. In addition, the design of high speed routers may be revised to take into account the Passport header.

10. RELATED WORK

Router Filters: This approach maintains filter tables at routers to discard spoofed packets, and does not modify packet headers. Ingress filtering [16], route-based filtering [26, 31], reverse path filtering [3], IDPF [14], and hop-count filtering [21] all fall into this category. As shown in § 8, ingress filtering provides little incentive for adoption. Route-based filtering involves non-trivial control overhead to update filter tables, and the control messages themselves need to be signed [26]. Reverse path filtering does not completely prevent address spoofing with asymmetric routing. IDPF binds a source address prefix to all incoming interfaces from which the prefix advertisement is received. It allows spoofing if an attacker's packets come from one of those interfaces, and does not work with asymmetric routing if an AS does not announce its prefixes to all providers. Hop-count filtering uses TTL heuristics to identify packets with spoofed source addresses, but attackers can still forge packets with spoofed addresses if they are no further from a destination than the sources they spoof.

Path Marking: This approach uses router-inserted path identifiers to approximate source locators [2, 45, 48, 49]. Unlike Passport, downstream routers cannot validate the authenticity of path identifiers stamped by upstream routers.

Traceback: Various traceback proposals aim to discover the sources of attack packets from router marks [38, 41, 47], or router state [40], or control messages [6]. Traceback may discover packet sources after packets are received, but does not detect spoofed packets at forwarding time like Passport.

Challenge-Response: IP or overlay routers can use connection cookies [9, 24] to validate the source address of a connection before forwarding a connection setup packet to end systems. However, this mechanism does not prevent spoofed packets from congesting a link before they reach the cookie generator.

Other cryptographic approaches. These approaches inserts secrets into packets to authenticate the source address. Spoofing Prevent Method (SPM) [4] uses a 32-bit secret key shared between a source and a destination AS to authenticate the source address of a packet, and is vulnerable to eavesdropper attacks. Passport differs from SPM in that it includes a key distribution protocol (while SPM does not), and uses keyed-MACs rather than plaintext keys so they cannot be transferred to other paths, and provides the defense-in-depth needed for DoS prevention by enabling ASes in the network to verify Passport headers. We have also implemented and evaluated Passport.

Perlman's work on sabotage-proof routing uses public-key digital signatures [33] to authenticate packet sources. Our design is geared to use more efficient symmetric key MACs and hence differs in many respects.

Visa [15], SIFF [46], TVA [49], the ticket system [32], Fastpass [43], and Platypus [35] use secrets in packet headers as capabilities to authorize packets to reach a destination. Passport uses secrets to authenticate the source addresses. Capability based systems can use Passport to verify the source addresses of the capability requests.

An early design of Passport was presented in [27]. This work improves [27] and makes it more practical by considering deployment issues and reducing unnecessary features for simplicity. It also provides an implementation, evaluation, and modeling study of Passport.

11. CONCLUSION

We present and evaluate Passport, a system that allows source addresses to be validated within the network. Passport uses efficient symmetric-key cryptography to place tokens on packets that allow each AS on the path to verify that a source address is valid. ASes obtain the symmetric keys via a Diffie-Hellman key exchange piggybacked in routing messages. Passport is incrementally deployable without upgrading hosts. We have implemented Passport, evaluated it, and run experiments on the Deterlab to demonstrate its usefulness in mitigating reflector attacks. We have also analyzed its security assurances and modeled its adoptability. Our performance evaluation shows that a software PC-based router can stamp or verify Passport headers at up to 2Gbps assuming an average packet size of 400 bytes [1]. Our adoptability modeling shows that Passport provides strong security benefits to drive its adoption, and is more adoptable than ingress filtering [16] and route-based filtering [26, 31]. Together, these results suggest that cryptography-based source address authentication is feasible and advantageous.

Acknowledgment

We thank Theodore Krovetz for his help on UMAC security, and Adrian Perrig and Debabrata Dash for providing the test topology for our adoptability study, and Nanogers and Steve Bellovin for answering questions on router clock synchronization, and anonymous reviewers and other members in our group for valuable feedback.

References

[1] CAIDA Report. `http://www.caida.org/analysis/AIX/plen_hist/`, 2000.

[2] K. Argyraki and D. Cheriton. Active Internet Traffic Filtering: Real-Time Response to Denial-of-Service Attacks. In *USENIX*, 2005.

[3] F. Baker and P. Savola. Ingress Filtering for Multihomed Networks. RFC 3704, 2004.

[4] A. B. Barr and H. Levy. Spoofing Prevention Method. In *IEEE Infocom*, 2005.

[5] T. Bates, E. Chen, and R. Chandra. BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP). RFC 4456, 2006.

[6] S. Bellovin. ICMP Traceback Messages, 2000. draft-bellovin-itrace-00.txt.

[7] R. Beverly and S. Bauer. The Spoofer Project: Inferring the Extent of Source Address Filtering on the Internet. In *USENIX SRUTI*, 2005.

[8] J. Black, S. Halevi, H. Krawczyk, T. Krovetz, and P. Rogaway. UMAC: Fast and Secure Message Authentication. In *Advances in Cryptology - CRYPTO '99*, 1999.

[9] M. Casado, A. Akella, P. Cao, N. Provos, and S. Shenker. Cookies Along Trust-Boundaries (CAT): Accurate and Deployable Flood Protection. In *USENIX SRUTI*, 2006.

[10] H. Chan, D. Dash, A. Perrig, and H. Zhang. Modeling Adoptability of Secure BGP Protocols. In *ACM SIGCOMM*, 2006.

[11] Deterlab. `http://www.deterlab.net/`.

[12] W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 1976.

[13] D.Magoni and J.J.Pansiot. Analysis of the Autonomous System Network Topology. *ACM SIGCOMM CCR*, 31(3), July 2001.

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

[15] D. Estrin, J. C. Mogul, G. Tsudik., and K.Anand. Visa Protocols for Controlling Inter-Organization Datagram Flow. *IEEE JSAC*, 1989.

[16] P. Ferguson and D. Senie. Network Ingress Filtering: Defeating Denial of Service Attacks which Employ IP Source Address Spoofing. RFC 2827, May 2000.

[17] A. Greenhalgh, M. Handley, and F. Huici. Using routing and tunneling to combat DoS attacks. In *USENIX SRUTI*, 2005.

[18] M. Handley, E. Kohler, A. Ghosh, O. Hodson, and P. Radoslavov. Designing Extensible IP Router Software. In *USENIX NSDI*, 2005.

[19] J. Hawkinson and T. Bates. Guidelines for creation, selection, and registration of an Autonomous System (AS). RFC 1930, 1996.

[20] Helion Technology. AES Cores. `http://www.heliontech.com/aes.htm`, 2008.

[21] C. Jin, H. Wang, and K. G. Shin. Hop-Count Filtering: An Effective Defense Against Spoofed DDoS Traffic. In *ACM CCS*, 2003.

[22] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The Click Modular Router. *ACM TOCS*, 18(3), Aug. 2000.

[23] T. Krovetz. UMAC: Message Authentication Code using Universal Hashing. RFC 4418, 2006.

[24] K. Lakshminarayanan, D. Adkins, A. Perrig, and I. Stoica. Taming IP Packet Flooding Attacks. In *Proc. HotNets-II*, 2003.

[25] A. K. Lenstra and E. R. Verheul. Selecting Cryptographic Key Sizes. *Lecture Notes in Computer Science*, 1751:446–265, 2000.

[26] J. Li, J. Mirkovic, M. Wang, P. Reiher, and L. Zhang. SAVE: Source Address Validity Enforcement. In *IEEE INFOCOM*, 2002.

[27] X. Liu, X. Yang, D. Wetherall, and T. Anderson. Efficient and Secure Source Authentication with Packet Passports. In *USENIX SRUTI*, 2006.

[28] Z. M. Mao, J. Rexford, J. Wang, and R. Katz. Towards an Accurate AS-Level Traceroute Tool. In *ACM SIGCOMM*, 2003.

[29] K. Nichols, S. Blake, F. Baker, and D. Black. Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. RFC 2474, 1998.

[30] E. Nordmark. Stateless IP/ICMP Translation Algorithm (SIIT). RFC 2765, 2000.

[31] K. Park and H. Lee. On the Effectiveness of Route-Based Packet Filtering for Distributed DoS Attack Prevention in Power-Law Internets. In *ACM SIGCOMM*, 2001.

[32] B. Patel and J. Crowcroft. Ticket Based Service Access for the Mobile User. In *ACM MobiCom*, 1997.

[33] R. Perlman. Network Layer Protocols with Byzantine Robustness. MIT Ph.D. Thesis, 1988.

[34] A. Perrig, R. Canetti, D. Song, and J. D. Tygar. Efficient and Secure Source Authentication for Multicast. In *NDSS*, 2001.

[35] B. Raghavan and A. C. Snoeren. A System for Authenticated Policy-Compliant Routing. In *ACM SIGCOMM*, 2004.

[36] Y. Rekhter, T. Li, and S. Hares. A Border Gateway Protocol 4 (BGP-4). RFC 4271, 2006.

[37] RouteViews Project. `http://www.routeviews.org/`.

[38] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Practical Network Support for IP Traceback. In *ACM SIGCOMM*, 2000.

[39] F. Scalzo. Anatomy of recent DNS reflector attacks from the victim and reflector points of view. Nanog37 Presentation, June 2006.

[40] A. Snoeren, C. Partridge, L. Sanchez, C. Jones, F. Tchakountio, S. Kent, and W. Strayer. Hash-Based IP Traceback. In *ACM SIGCOMM*, 2001.

[41] D. Song and A. Perrig. Advance and Authenticated Marking Schemes for IP Traceback. In *Proc. IEEE Infocom*, 2001.

[42] R. Thomas. ISP Security BOF, NANOG 28. `http://www.nanog.org/mtg-0306/pdf/thomas.pdf`, 2003.

[43] D. Wendlandt, D. G. Andersen, and A. Perrig. FastPass: Providing First-Packet Delivery. Technical report, CMU-CyLab, 2006.

[44] J. Winick and S. Jamin. Inet-3.0: Internet Topology Generator. Technical Report CSE-TR-456-02, University of Michigan, 2002.

[45] A. Yaar, A. Perrig, and D. Song. Pi: A Path Identification Mechanism to Defend Against DDoS Attacks. In *IEEE Symposium on Security and Privacy*, 2003.

[46] 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.

[47] A. Yaar, A. Perrig, and D. Song. FIT: Fast Internet Traceback. In *Proc. of IEEE Infocom*, 2005.

[48] A. Yaar, A. Perrig, and D. Song. StackPi: New Packet Marking and Filtering Mechanisms for DDoS and IP Spoofing Defense. *IEEE JSAC*, 2006.

[49] X. Yang, D. Wetherall, and T. Anderson. A DoS-Limiting Network Architecture. In *ACM SIGCOMM*, 2005.

[50] K. Zetter. Cisco Security Hole a Whopper. `http://www.wired.com/politics/security/news/2005/07/68328`, 2005.

[51] X. Zhao, D. Pei, L. Wang, D. Massey, A. Mankin, S. F. Wu, and L. Zhang. An Analysis of BGP Multiple Origin AS (MOAS) Conflicts. In *ACM IMW*, 2001.