Optimizing BGP Security by Exploiting Path Stability

Kevin Butler[o], Patrick McDaniel, and William Aiello†
SIIS Laboratory, Computer Science and Engineering
The Pennsylvania State University
University Park PA, 16802, USA
†Department of Computer Science
University of British Columbia
Vancouver BC, V6T 1Z4, Canada

[o]*Corresponding author. Contact info:*
Phone: +1 (814) 865-6245
Fax: +1 (814) 865-3176
Email: butler@cse.psu.edu

November 19, 2008

Abstract

The Border Gateway Protocol (BGP) is the *de facto* interdomain routing protocol on the Internet. While the serious vulnerabilities of BGP are well known, no security solution has been widely deployed. The lack of adoption is largely caused by a failure to find a balance between deployability, cost, and security. In this paper, we consider the design and performance of BGP path authentication constructions that limit resource costs by exploiting route stability. Based on a year-long study of BGP traffic and indirectly supported by findings within the networking community, we observe that routing paths are highly stable. This observation leads to comprehensive and efficient constructions for path authentication. We empirically analyze the resource consumption of the proposed constructions via trace-based simulations. This latter study indicates that our constructions can reduce validation costs by as much as 97.3% over existing proposals while requiring nominal storage resources. We conclude by considering operational issues related to incremental deployment of our solution.

Keywords: routing, security, path stability, BGP

1 Introduction

The Border Gateway Protocol (BGP) [39, 38] is the dominant interdomain routing protocol on the Internet. BGP establishes and maintains associations between IP address *prefixes* [35] (addresses) and source specific paths to the autonomous systems (networks) in which they reside. Each AS selects the best paths based on the advertised paths and routing policy. However, the BGP protocol is largely devoid of any security [34, 43, 1, 27, 32, 4]. One critical vulnerability resulting from this lack of security allows an adversary to manipulate *paths*, adversely affecting the ways that destinations are routed to. There are ancillary attacks that are as dangerous as those which attack routing path selection. Because of a lack of security services, adversaries are free to subvert the Internet interdomain routing infrastructure and through it, manipulate the underlying IP traffic. For example, addresses can be made unavailable (by mounting a denial of service attack), and traffic can be rerouted through malicious networks and monitored (an attack on traffic confidentiality), or directly altered (an attack on integrity). Because these attacks threaten the basic transport on which all network communication relies, many regard this as one of the most vital security vulnerabilities on the Internet.

While many approaches have been proposed to address BGP security [43, 17, 30, 47, 5, 8, 18, 45, 44, 46, 52], none have been widely deployed. The lack of adoption is largely caused by a failure by the community to find an acceptable balance between cost and security. For example, the S-BGP protocol [17] offers comprehensive security by authenticating routing artifacts (e.g., prefix and path advertisements, withdrawals, etc.) using asymmetric cryptography. However, the computational and storage costs of performing strong S-BGP style authentication are viewed to be prohibitive in many environments [5, 31, 9, 46]. Recent works in BGP have sought optimizations that reduce these costs. For example, among others, these works have used advanced cryptography [8, 9], out-of-band security [5], relaxed guarantees [30], or address usage patterns [21] to reduce security costs. In recent work, Zhao et al. exploited the structure of the BGP protocol to implement an AS-local optimization for path validation [52]. This tuning of cryptography to match protocol behavior in conjunction with advanced cryptographic techniques was notable in that it significantly reduced costs over techniques like S-BGP. This work takes a similar approach, but instead attempts to exploit the *global* use of BGP.

We argue throughout that optimizations that flow from common protocol use are most likely to be effective. Our analysis of a year of global BGP traffic, supported indirectly by studies of BGP traffic [46, 3,

29, 22, 40], indicates a large degree of *path stability*. That is, most paths remain stable for extended periods and few "new" paths are seen. We study the 40 Route Views listening points [25], and found that in the average case, less than 2% of prefixes were advertised using more than 10 paths, and less than 0.06% were advertised with more than 20 paths during a single month.

The observed limited diversity and high stability of BGP paths allows us to explore a range of efficient cryptographic structures for path authentication. We develop a formal model of path authentication in BGP. We define and prove the security of our novel and efficient solutions under this model. Our authentication proof systems indicate the set of paths that an AS will announce during its lifetime. Associated with each path is a collection of *succinct validation proofs*, represented as *tokens* and released by the AS as evidence of its authenticity. Token validation is amortized over all proofs associated with that prefix. ASes using these constructions create long-lifetime cryptographic proof systems [24, 28] that validate all paths that they are likely to advertise. Efficiently validatable tokens reflecting current best paths are derived from these proof systems and distributed throughout the Internet. In this way, the costs of heavyweight cryptographic operations are amortized over many validations.

We further compare the computational cost of our solutions against other BGP security solutions via trace-based simulation. Our simulations demonstrate that our techniques reduce the costs through reducing signature validations by up to 97.3% over proposed solutions, and the storage costs at validating ASes are nominal. Note that schemes such as signature amortization, proposed by Zhao et al., and SPV [9], optimize BGP in ways orthogonal to our solutions, and incorporating them could lead to even greater reductions in computational costs. However, we defer the analysis of the joint advantage of these solutions to future work.

We begin in the following section by outlining the operation and security requirements of BGP.

2 Interdomain Routing

BGP provides two essential services[1]: the mapping of address prefixes (e.g., `192.168.0.0/16`) onto the ASes that own them, and the construction of source specific paths to each reachable prefix. The interdomain routing topology is defined by physical links between adjacent ASes. Each AS *originates* the prefixes associated with a network by identifying and enumerating them in an UPDATE message sent to its neighbors

[1]Throughout we refer to the AS to AS communication protocol *eBGP* generically as BGP. The intra-AS *iBGP* protocol governs the way in which eBGP speaking edge-routers within an AS exchange routing information. iBGP is explictly outside the scope of this work.

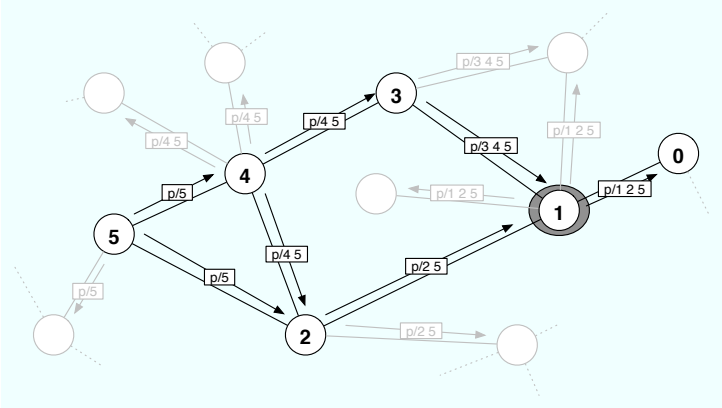

Figure 1: BGP Path discovery - AS5 originates the prefix p by announcing it to its neighbors (e.g., AS4). AS4 further propagates the prefix to its neighbors AS2 and AS3 after prepending its AS number to the prefix. AS1 (highlighted) receives routes from AS2 and AS3, and selects the best route ($\{2\ 5\}$), which is then propagated further (to AS0 and others).

(adjacent ASes). Received announcements are recursively concatenated with local AS numbers [11] and propagated, AS by AS, to form a routing path. This path (also called a *route*) is used to forward network traffic to the origin. Note that an AS may receive many paths for a single prefix. The AS identifies the "best" path using the *path selection algorithm*. The selection algorithm determines the best route by evaluating path length, policy, and other factors. Only the selected best path is propagated. IP traffic is routed, hop-by-hop, based on the best path known by the AS. Figure 1 illustrates route advertisement and path selection.

Which route represents the best path is re-evaluated each time a new route for a prefix is received. Suppression of non-best routes prevents undesirable routes from polluting the larger Internet, and is a key ingredient to the scalability of BGP. Recursive propagation of best routes ensures that every AS on the Internet acquires a route for every reachable prefix. A route is *withdrawn* when the AS discovers that the prefix is no longer reachable.

The ubiquity of BGP is also one of its greatest weaknesses. The number of ASes and complexity of their interaction affords an adversary opportunities to monitor, disrupt, or manipulate the routing process. The Routing Protocol Security (rpsec) working group of the IETF postulate a universe of possible effects of routing vulnerabilities [2]. Traffic congestion, black-holing, routing loops, slowed or prevented convergence, instability, traffic eavesdropping, network partitioning, and increased delay were deemed the most damaging consequences. The group's analysis led to a statement of general routing security requirements [37], and more specifically, to requirements for BGP security [27]. We consider the vulnerabilities germane to current

work below and review broader classes of vulnerabilities and solutions in Section 7.

BGP security concerns are often classified by the three broad categories of data exchanged [32, 4]: signaling, prefix origins, and paths. Attacks on BGP signaling frustrate the session by incorrectly reporting errors, masquerading as other entities, or by consuming the victim's resources [27]. Authenticating an AS's right to advertise (originate) a prefix is essential to securing BGP. Failure to perform this authentication invites prefix hijacking: an adversary can steal address space simply by advertising it [17, 42, 21].

This paper investigates *path authentication*. Hu et al. identified the following classes of path attacks [9]: (a) *path forgery* - the adversary may attempt to forge paths in order to influence packet routing, (b) *path modification* - an adversary may add, remove, or alter data in the path or policy, (c) *denial of service* - an adversary consumes a victim's resources by sending spurious routes, and (d) *worm-holing* - in which colluding adversaries create false AS to AS links. Note that the first two classes are attacks, whereas the second two could be more accurately classified as consequences. Moreover, *worm-holing* is less of an attack on paths, but more of an attack on the topology. The false topology generated can be used to introduce incorrect paths, even if a path validation approach is perfectly implemented and deployed. With the exception of soBGP [30] (see Section 7), few security proposals address worm-holing, as it requires validation of BGP peering.

If an adversary can forge or modify routes, then it can *black-hole* traffic routed to it. To accomplish this, the adversary announces a highly desirable route that is incident to the path, e.g., by advertising a very short path. Traffic flowing to that prefix will be routed to the adversary and filtered. If the adversary wants to destabilize the network while remaining relatively clandestine, it can randomly drop a percentage of the traffic (called *grey-holing*). Note that it takes few drops to vastly reduce the throughput between the victim and the destination: each drop causes the congestion control algorithm to aggressively throttle traffic [36]. Connection recovery is slow, and the attacker gains advantage with little effort [50]. Paths may also be manipulated to route traffic through malicious ASes for monitoring [4]. That is, if an adversary can redirect traffic (as above), then it can monitor, record, or even modify that traffic as it transits its network. Furthermore, an AS's ability to filter or rapidly advertise and withdraw advertisements leads to a range of DoS attacks [27, 49] that may easily render targeted networks unreachable.

3 Path Validation Constructions and Formalisms

In this section, we define what we mean by attestations and route attestation tags and formally state their security properties. Route attestation tags are very similar to route attestations as defined in [17, 16, 15] with several minor differences highlighted in this section. We assume that the reader has a general familiarity with cryptographic primitives such as hash chains, hash tables and digital signatures. These constructions are explored in greater detail later in the section. We begin in the following subsection with a brief overview of our approach to path authentication, and continue with a formal description of its semantics, operation, and security.

3.1 Path Validation Constructions

We begin by introducing constructions for path authentication that will be subsequently examined and evaluated throughout the rest of the paper. As indicated in the previous section, any solution that secures the path must provide at least the following simplified guarantees: an AS receiving a route must be able to a) authenticate the source of an advertisement, b) authenticate that the ASes in the path advertised the sub-paths in the order which they are listed (i.e., no ASes were added or removed), and c) validate the times at which each of the (sub)advertisements occurred. Note that in reality the security guarantees are somewhat more subtle, but these definitions are sufficient to motivate the following discussion.

Consider S-BGP attestations [17]. As shown in figure 1, a BGP speaker sending a route announcement to its peer signs each announcement as it propagates across the network. If the path to a given network prefix changes, a new announcement is signed and sent to the peer. For example, assume that prefix P, originated by AS 5, is being advertised by AS 1, which knows two paths to the destination: $\{2\ 5\}$ and $\{3\ 4\ 5\}$[2]. In the first time period, the attestation issued by AS 1 will be

$$[P, \{2\ 5\}, t_1]_{S_2}$$

If, in the next time period, there is a link failure between AS 5 and AS 2, AS 2 will readvertise its optimal route as $\{2\ 4\ 5\}$. AS 1 will now arbitrarily pick between this route and $\{3\ 4\ 5\}$. If we assume that $\{3\ 4\ 5\}$ is chosen then the attestation in this time period will be

[2] We apply the convention that paths grow from right to left, with the originating AS occupying the rightmost value in the path vector.

$$[P, \{3\ 4\ 5\}, t_2]_{S_3}$$

If there is a link problem between AS 3 and AS 4 in the next time period, the route advertised by AS 1 will now be

$$[P, \{2\ 4\ 5\}, t_3]_{S_2}$$

where S_n represents a digital signature issued for the route attestation by AS n. The signature authenticates AS 1 as being the verifiable source of the announcement. S-BGP announcements are recursively signed: signed attestation proves not only that the peer vouches for the path, but that each hop in the path also vouches for the included sub-path. For example, assume at time t_n that AS 1 receives the path $\{2\ 5\}$; in reality, the received attestation will have the logical form

$$[[[P, \{5\}, t_{n-2}]_{S_5}]P, \{2\ 5\}, t_{n-1}]_{S_2}$$

as originating AS 5 initially signs the path and AS 2 signs that original attestation and itself as part of the path vector for prefix P. When AS 1 advertises this route, it will cumulatively sign over the other attestations and the new path vector, as follows:

$$[[[[[P, \{5\}, t_{n-2}]_{S_5}]P, \{2\ 5\}, t_{n-1}]_{S_2}]P, \{1\ 2\ 5\}, t_n]_{S_1}$$

In this manner, the path can be recursively verified by validating each AS path signature back to the route origin. Finally, because the timestamp of each announcement is included, replay attacks are avoided. Thus, S-BGP attestations meet the requirements for path authentication.

It is obvious to see that these attestations can be costly in practice: there are currently over 200,000 prefixes being advertised by 22,000 ASes in the Internet [10]. This can lead to huge numbers of signatures and validations at each AS. We now introduce several novel approaches that attempt to mitigate these costs.

One opportunity to optimize cost is through signature aggregation. For example, we can exploit the fact that paths are stable: only a few paths are likely to be advertised for most prefixes. We propose that a *hash chain* [20] be initially generated for each distinct path associated with a particular prefix. The first value generated for each path is sent to the peer, with the entire message signed. An *authentication token*

consisting of the next value in the hash chain for the new path is sent to the peer whenever a different route is advertised. The peer hashes the token forward to verify that it arrives at the anchor value of the hash chain. Hashing is approximately three orders of magnitude faster than a signature validation in software [9]. Thus, validation costs are greatly reduced. When the hash chain has been exhausted, a new announcement containing all paths and the signed tokens is sent to the BGP peer. Returning to the previous example, AS 1 sends all its paths in a single list, along with the tokens representing the hash chain anchors as follows:

$$\left[\begin{array}{l} P, \{2\ 5\}, h^{365}(x_1) \\ P, \{2\ 4\ 5\}, h^{365}(x_2) \\ P, \{3\ 4\ 5\}, h^{365}(x_3) \\ P, \{3\ 4\ 2\ 5\}, h^{365}(x_4) \end{array}\right]_{S_1}$$

where x_1, x_2, x_3, and x_4 are the randomly-generated seed values for the hash chains for the paths $\{2\ 5\}$, $\{2\ 4\ 5\}$, $\{3\ 4\ 5\}$ and $\{3\ 4\ 2\ 5\}$, respectively, $h^n(x)$ represents a hash chain of length n with seed x, and the hash chain length of 365 is an example construction parameter, e.g., representing a chain that generates a token once a day for a year.[3] At time t_n, the authentication token associated with that time period is sent that represents the route advertised at that time:

$$t_1 \rightarrow h^{365-1}(x_1)$$
$$t_2 \rightarrow h^{365-2}(x_2)$$
$$t_3 \rightarrow h^{365-3}(x_3)$$

The token provides replay protection due to the infeasibility of generating a token representing a later time value. Note that these three authentication tokens fulfill the same security guarantees as their equivalent S-BGP attestations, i.e., $t_1 \rightarrow h^{365-1}(x_1)$ has the equivalent security guarantees to $[P\{2\ 5\}, t_1]_{S_2}$, etc. There is a minor security loss that is contingent on the size of the construction parameter. Because a signature is only generated when the hash chain is exhausted, a malicious peer can advertise any of the paths sent in the aggregate signature with the appropriate authentication token, regardless of whether it is optimal. There

[3] AS 1 also transmits lists received from each previous peer and onion-signs those attestations as in the S-BGP example. We omit the full details for clarity.

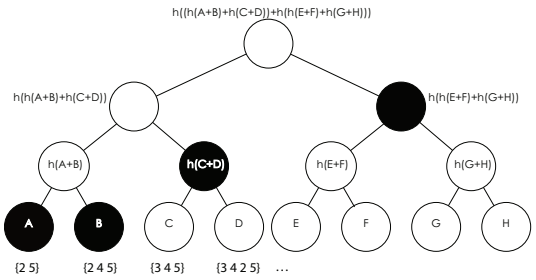

Figure 2: Tree construction for path aggregation. As in the list construction, we assume that A has been selected as representing the optimal path. Only the black nodes are hashed, and only the root is signed. There are $\lceil \log_2 n \rceil$ hashes that need to be computed for n leaves of the tree.

is no validation of routes until the next signature is generated. For example, a peer advertising path a and subsequently advertising path b can again advertise a even if it has been withdrawn by an upstream peer. However, a peer can always suppress an advertisement with any variant of BGP; the additional threat posed by an attacker advertising a pre-existing, validated path is minimal. The window for these threats can be reduced by making the construction parameter smaller, at the cost of having to generate signatures more frequently.

An advertising AS forwards not only the authentication tokens for its advertisement, but also tokens it received for the included sub-paths. This provides both verification of the peer announcement and recursive validation of all encompassed announcements back to the origin AS. In this way, the approach achieves similar security guarantees to that of S-BGP attestation: paths cannot be forged, sub-routes can be validated, and the timing of the announcements can be validated.

While the preceding construction mitigates the computational costs of recursively signed advertisements, it introduces other resource costs. Because each signed list contains all paths associated with a given prefix, the bandwidth and storage costs associated with processing these lists may be prohibitive. For example, current routers have exceedingly small amounts of available main memory [26], and hard-disks induce considerably higher access latencies and often fall victim to more frequent failures.

As has been demonstrated in many domains, transmission and storage costs associated with authenticated material can be mitigated by using cryptographic proof systems, e.g., Merkle hash trees [24] and authenticated dictionaries [28, 6]. Our tree path authentication construction is based on the Merkle hash tree. In this construction, a *succinct* set-membership proof is generated by the announcing peer; as shown in Figure 2, each advertised path forms a leaf in the Merkle tree. When a path is announced, only a hash of the leaf's sibling, the parent's sibling, etc., up to the root node, are required. The root of the tree is signed. The

computational costs are slightly greater than with the list construction, as a number of hashes proportional to the height of the tree must be computed by the peer receiving the path announcement; however, because of the very low cost of hashing, the extra effort is minimal. Hence, the tree construction provides an attractive balance between computational, storage, and bandwidth costs. Generation of hash chains for paths follows the same process as with the list construction; the leaves of the hash tree are associated with the generated hash tree value, such that only the authentication token is necessary to be sent for each route announcement.

We now consider a number of alternate constructions based on *how* paths are aggregated. In each case, hash chains are generated for each path as described above. However, we construct a different tree whose structure relates to how aggregation is performed, to further amortize costs and exploit different computational and storage trade-offs. We refer to the approach described above as the *prefix* scheme: the AS constructs a tree where the leaves represent all of the distinct paths it advertises for that prefix. By contrast, in the *origin* scheme, the AS constructs a tree where the leaves represent all of the distinct (prefix,paths) pairs it advertises with the given origin AS. The *all* AS construction creates a single tree where the leaves represent all the paths the AS historically advertises. One can view these approaches as simply as different partionings of the paths an AS may advertise: the *prefix* scheme creates a tree with all the paths for each unique prefix, the *origin* scheme creates a tree for every unique AS that originates a prefix, and the *all* creates a single tree for all the advertisements that the AS emits.

Note that in both the list and tree structures, if a new announcement is received that contains a previously unseen new path, representing a new path to a given prefix, the announcement is sent to peers as an S-BGP type route attestation. When the aggregate constructions are resent to peers, this new path will be part of the aggregation. Hence, in the degenerate case where all advertised paths are new, the scheme reverts to S-BGP style advertisements.

We consider the stability of path advertisements in section 4. The degree to which paths are stable will determine how well the optimizations perform in practice. We explore the computational overheads of our proposed approaches via trace-based simulation in section 5. While our scheme does not explicitly address optimizations such as enumerating peer routes with bit vectors as shown in [52], as they are orthogonal to our goals, such methods can be employed to further reduce validation costs.

3.2 Attestations and Route Attestation Tags

In previous works, route attestations were defined as a sequence of statements signed by routers using public key signatures. Our route attestation tags are also a sequence of attestations by routers, but here we allow the attestations to be more general public key authentication methods. In particular, an attestation may be either a signature or a *set-membership proof*. Set-membership proofs are essentially signatures of Merkle hash trees [24].

We first state several definitions that will be used below. We then state formally the definition of a set-membership proof, its definition of security, and several examples. Next, we define a *route attestation tag* or RAT as a sequence of attestations. Finally, we describe our scheme, as well as the schemes of S-BGP [17] and Nicol et al.[31] as instantiations of the general set-up. These descriptions are used in the subsequent sections, where the performance tradeoffs of these schemes are empirically analyzed.

Note that Zhao et al. [52] incorportate the signature amortization scheme from Nicol et al., but also include aggregate signatures as an additional optimization. Aggregate signatures can shorten the size of signatures and reduce memory footprints, but do not aid in relieving the computational burden of signatures and validations; accordingly, we strictly consider the signature-amortization element of this work. Our scheme may also benefit in terms of memory usage from aggregate signatures, but the implication of their use is left for future work.

The formalization below is general enough to capture not only our proposed schemes but the S-BGP scheme and the scheme of Nicol et al. as well. At the same time, it is specific enough to allow for a precise definition of existential forgery of a route announcement and a reduction to the security of standard cryptographic primitives. Most of the material in this section is based on prior work but the impetus is to formalize the ideas sufficiently to present formal definitions of validity and proofs of security. As seen repeatedly in security research, there is value in having formal definitions of security and using them to reduce the security of one primitive to another.

3.3 Notation

Let $\mathcal{ASN} = \{1, \ldots, 2^{16} - 1\}$ be the set of all unique identifiers for an Autonomous System. These are the so-called Autonomous System Numbers. An *AS path* is a sequence, possibly empty, of AS numbers. Given a path $p \in \mathcal{ASN}^*$, let p_i, $i \geq 1$, denote the ith element in the sequence. Furthermore, let $p_i^{\leq}$, $i \geq 1$,

denote the subsequence of the first i elements of p. For example, if $p = (23, 1708, 229)$, then $p_2 = 1708$ and $p_2^{\leq} = (23, 1708)$.

In BGP, AS padding is allowed. That is, a legitimate AS path can have a sequence of consecutive values that are identical. This is equivalent to saying that BGP allows paths with self loops (but not other kinds of loops). We call a path *almost simple* if it has no loops except for self loops.

Let $G = (\mathcal{ASN}, \mathcal{E})$ denote the AS graph. A pair of AS numbers (a_1, a_2) is in $\mathcal{E}$ if AS a_1 and AS a_2 have a service level agreement (SLA) to be eBGP neighbors. Note that $\mathcal{E}$ does not capture which pairs of ASes have active eBGP sessions between routers at the current time. That is, if (a_1, a_2) is in $\mathcal{E}$, there may be no current eBGP session between a router in a_1 and a router in a_2. Nonetheless, the edge in the graph G is maintained as long as the neighbors have an eBGP SLA. A path p is denoted *topology respecting* if every edge in the path is also an edge in G.

A route is a pair consisting of an address block and an AS path. Given an address block b and a path $p = (a_1, a_2, \ldots, a_k)$, the route r for b and p is written as $r = (b, p)$ or as $r = (b; a_1, a_2, \cdots, a_k)$. Considering the latter as a sequence, r_i, $i \geq 0$, denotes the ith element of the sequence and $r_i^{\leq}$ denotes the subsequence of r from r_0 to r_i, inclusive. Note that $r_0 = b$, and that for $i \geq 1$, $r_i = (b, p_i)$, and $r_i^{\leq} = (b, p_i^{\leq})$. These definitions will be useful when defining the cryptographic mechanisms for protecting entire routes.

3.4 Signatures and Set-Membership Proofs

For completeness, recall the definition of a signature scheme. A signature scheme consists of three functions:

1. a randomized generation algorithm G that takes as input a security parameter (e.g., the desired length of the output)and generates a public/private key pair $(\mathsf{pk}, \mathsf{sk})$;
2. a signing algorithm S that takes as input a secret key sk and a value a and computes a signature σ; and,
3. a verification algorithm V that takes as input a public key pk, a value a, and a signature σ and outputs "accept" or "reject".

G, S, and V satisfy the following signature-correctness condition. For all $(\mathsf{pk}, \mathsf{sk})$ generated by G and all strings a, if $\sigma = S(\mathsf{sk}, a)$, then $V(\mathsf{pk}, a, \sigma) =$"accept".

Before defining set-membership proofs, we first give the motivation and intuition. We first note that the "signer" for a set-membership proof must commit to the values of the set ahead of time. But having

done that, the signer needs only to perform one public key signature computation and that same signature is used to produce the set-membership proof for each element of the set. Likewise, any verifier needs only to perform one public key signature validation in order to validate all the membership proofs for the elements of the set. This can result in significant computational savings. In our setting, such savings are dependent upon whether the set of AS paths that a router in an AS announces over time is sufficiently stable that the router can optimistically commit to that set of paths (or a superset of its current "announcing" set) and not require announcing too many paths not in that set. The size and stability of sets of AS paths is studied carefully in the next section. A comparison of computational and bandwidth costs of several varieties of attestation follows in the subsequent section.

Definition 1. A membership proof scheme consists of a signature scheme (G, S, V), and three additional algorithms:

1. The set-tag function T takes a set $\mathcal{A}$ and a random string r, and produces two values: $T_0 = \tau$, denoted the set tag, and $T_1 = k_\tau$, denoted the set-tag key.
2. The membership-proof algorithm P takes as input an element a and a key k, and outputs a string π, denoted the membership proof.
3. The set-tag-extraction function E takes as input an element a and a string π, and outputs a string τ.

T, P, and E satisfy the following correctness condition. For all r, all sets $\mathcal{A}$, and each element a of $\mathcal{A}$, if $T(\mathcal{A}, r) = \tau, k$, and $P(a, k) = \pi$, then $E(a, \pi) = \tau$. In other words, this says that for each element of a set, if a membership proof for the element is created using a legitimate set-tag key for that set, then the string extracted from that membership proof will be the legitimate set tag for the set.

A signer uses a membership-proof scheme as follows. For a given set $\mathcal{A}$, the signer first computes $T(\mathcal{A}, r) = \tau, k_\tau$. It then computes $S(\mathsf{sk}, \tau) = \sigma$. At that point it can make $(\mathcal{A}, \tau, \sigma)$ public. It stores (τ, k_τ) in memory (i.e., k_τ remains secret). Given a set tag τ and an element a, where τ is a set tag previously computed, the signer creates a membership proof by first retrieving k_τ from memory and computing the membership proof $P(a, k_\tau) = \pi$. The signer then makes a and π public.

A verifier uses a membership-proof scheme as follows. Given the public key of the signer and an element a, a purported membership proof π, and a set tag and its purported signature (τ, σ), the verifier first computes $E(a, \pi)$ and verifies it is equal to τ, then runs the signature verification algorithm $V(\mathsf{pk}, \tau, \sigma)$. The verifier accepts if V accepts.

Definition 2. A signature scheme is (k, T, ϵ) secure against existential forgery if it also satisfies the following security requirement. An adversary is allowed to see k (message, signature) pairs where the messages can be chosen by the adversary adaptively. No adversary running in time at most T can generate a message, signature pair that passes the verification except with probability at most ϵ.

A set-membership proof is defined similarly. It consists of three functions, G', S', and V': 1) a randomized key generation algorithm G' takes as input a security parameter (e.g., the desired length of the output) and a set of elements, A, and generates a public/private key pair $(\mathsf{pk}', \mathsf{sk}')$; 2) a signing algorithm S' that takes as input a secret key sk', a set A, and an element of the set a, and computes a set membership proof π; and, 3) a verification algorithm V' which takes as input a public key pk', a value a, and a proof π and outputs *accept* or *reject*. Note that if a is not in A then S' outputs $\perp$.

G', S', and V' satisfy the following correctness condition. For all A and all $(\mathsf{pk}', \mathsf{sk}')$ generated by G on A, and all strings $a \in A$, if $\pi = S'(\mathsf{sk}', A, a)$, then $V'(\mathsf{pk}', a, \pi) =$ "accept".

Definition 3. A set-membership proof scheme is (k, T, ϵ) secure against existential forgery if it also satisfies the following security requirement. An adversary is allowed to ask for public keys to be generated for sets of its choosing. The adversary is then allowed to see the signatures for k (set, set element) pairs where the pairs can be chosen by the adversary adaptively. No adversary running in time at most T can generate a (signature, set element, public key) triple that passes verification, except with probability at most ϵ.

The above definition of a set-membership proof scheme may be modified to include ancillary information about the set. That is, the signing algorithm may be modified to include this ancillary information about the set as input. If this is the case, the verification algorithm must be modified as well to include this ancillary information for proper verification.

A secure set-membership proof scheme can be constructed from a secure signature scheme and a hash function secure against second preimage attacks (for random domain elements). The advantage of a set membership proof scheme over a signature scheme is that in practice for both the signer and the verifier, the expensive public key computations need only be done once and then cached for any given set. [4] This efficiency comes at the price of larger space requirements but we note that the size of the membership proofs can be made logarithmic in the cardinality of the set. An example of a set membership proof system is the combination of Merkle hash trees and public key signatures as in the example above.

[4] This is not transparent from the abstract description of a set membership proof scheme above. A formal description of a set membership proof scheme that explicitly breaks out the public key computations is cumbersome and omitted here for brevity.

An important property of a set-membership proof scheme to highlight is that the signer only needs to compute T and S once, regardless of how many set elements it will eventually compute membership proofs for. That is, the cost of one public key signature computation can be amortized over the cost of many set-membership proofs. Likewise, a verifier needs only to run the signature verification algorithm on one valid (τ, σ) pair. It can cache the positive result using τ as a key. Subsequent membership proofs with set tag τ require only the verifier to run E, which is not a public key algorithm. Thus, the cost of one public key signature verification can be amortized over the cost of many set-membership verifications. In subsequent sections, we analyze the amortization savings that can be realized in practice on real BGP data streams.

Note that a membership proof scheme can be used as a signature scheme by setting $\mathcal{A}$ to be a singleton set.

We remark that an expiration time or validity period can be imposed on the membership proof system of a set by computing σ as a signature of both the set-tag and an expiration time or validity period, respectively.

It is well known that these membership proof schemes are secure in the sense above if the primitives on which they are based are secure; that is, if the signature is secure against existential forgery, the function f is one way, and the hash function is second preimage resistant.

3.5 Route Attestation Tags

An attestation by an identity x about a string α is denoted $A(x; \alpha)$. An attestation is either a secure signature signed by the secret key of x or it is a membership proof of α by the identity x (using the secret key of x). We will denote an attestation by x about a string β *to an identity* y by $A(x; \beta : y)$. This is just an attestation $A(x; \alpha)$ with $\alpha = \beta : y$. Attestations may also have timestamps or expiration times. These may be used, in part, as anti-replay mechanisms. For purposes of exposition, we do not include timestamps in the notation. We defer discussion of replay to later in this section.

Definition 4. For a given route we define a *route attestation tag* or RAT, as follows. A RAT takes as an input a route $r = (b, p)$. $RAT(r)$ is a sequence of attestations defined recursively as follows.

$$RAT(r_i^{\leq}) = RAT(r_{i-1}^{\leq}), A(p_{i-1}; r_i^{\leq} : p_i)$$

for $i = 2, ..., |p|$. The base case is $RAT(r_1^{\leq})$. This is the origin authentication tag, or OAT, for ownership of the address block $r_0 = b$ by the AS with identifier p_1. The semantics of $OAT(b, a)$ were discussed

extensively in [21]. Briefly, the $OAT(b, a)$ includes: a.) a chain of attestations from IANA to an organization O attesting to the fact that the ownership of the address block b has been delegated to O; b.) an attestation by IANA that it has assigned the AS identifier a to O; and c.) an attestation by O that it has assigned the address block b to AS a.

As an example, let $p = (a_1, a_2, a_3, a_4)$. Then

$$\begin{aligned} RAT(b; a_1, a_2, a_3, a_4) \quad = \quad & OAT(b, a_1), \\ & A(a_1; (b; a_1) : a_2), \\ & A(a_2; (b; a_1, a_2) : a_3), \\ & A(a_3; (b; a_1, a_2, a_3) : a_4) \end{aligned}$$

Note that the final attestation in $RAT(b; p)$ is by the second to last AS in the path, i.e., by AS $p_{|p|-1}$.

A RAT is valid only if all of the associated attestations are valid and the OAT is valid. Note that RATs as defined here are nearly identical to the definition of route attestations defined in [17]. The only minor differences are the inclusion of the origin authentication tag and the slight generalization to allow both signatures and set-membership proofs in the individual router attestations.

We denote the concatenation of a route $r = (b; p)$ and an AS a by $r.a$, where this is just the route given by the pair $(b; p.a)$, i.e., the path of r extended by one hop to a.

Definition 5. A route $r = (b, p)$, and an accompanying $RAT(r.a')$, when received in an update over an eBGP session by a router in AS a is considered valid only if:

1. $a = a'$,
2. $p.a$ is almost simple,
3. the RAT of $r.a$ is valid, i.e., the pair $(r.a, RAT(r.a))$ validates, and
4. the route was received over an authenticated eBGP session with a router in AS a^* where a^* must equal the last AS in the AS path p, i.e., $a^* = p_{|p|}$.

As defined above, a router that announces its new best AS path for a given address block to all of its neighbors must send a slightly different attestation to each of its eBGP neighbors. That is, to announce the

route r it must send $r, RAT(r.a)$ to an eBGP peer in AS a, and $r, RAT(r.a')$ to an eBGP peer in AS a' etc. At first glance this may seem unnecessary. However, different routers in the same AS may announce a different best AS path for the same prefix. If when advertising the route r, the router simply attested to the route up to and including its AS, it is easy to construct cases in which upstream routers can forge routes [32].

A similar reason argues for the requirement that the prefix be included in all of the attestations of a RAT. The alternative is to have the attestations in the RAT include only the AS path and to separately include the origin authentication tag for the prefix and origin AS. However, such a scheme allows for the following type of attack. Suppose a router in AS b receives routes for two different prefixes both originated by AS a, e.g, $(b; a.p.b)$ and $(b'; a.p'.b)$ and the origin authentication tags binding b and b' to a. If the attestations in the RATs contain the appropriate AS path prefixes but are not required to contain the address block, then the router in AS b can create RATs that will validate for routes it did not receive. In this example the router can create a valid RAT for $(b; a.p'.b)$ and $(b'; a.p.b)$, thus altering in an undetected fashion the routes for the prefixes b and b'.

Note that in order for a router in AS a to check the validity of $RAT(r.a)$, it is not sufficient for the router to simply have the certified value of the public key of its eBGP neighbor that sent it the route. The router must have the certified public keys of all ASes in order to check the attestations of each AS in the route. Here we assume a PKI provides each router with the certified public keys of all ASes. For a discussion of such a PKI see [42].

We now address the issue of the security guarantee provided by the RAT construction. Intuitively, we would like to say that as long as the attestation scheme used in a RAT is not existentially forgeable, then that RAT scheme is not existentially forgeable in the sense that an adversary cannot create a valid (route, RAT) pair that it has not previously seen. Unfortunately, it is not quite that simple. This is due to the fact that every AS, including malicious ones, are able to extract or extend valid $(r, RAT(r))$ pairs sent to them legitimately in several ways. For example, from a valid route attestation for r, it is easy to extract a valid route attestation tag for each prefix of r, i.e., $r_i^{\leq}$ for $i = 1, \cdots, |r|$. This follows directly from the recursive definition. As another example, if a router in AS a receives a valid pair $(r.a, RAT(r.a))$, then a (possibly different) router in a can compute a valid $RAT(r.a.a')$ for any neighboring AS a'. This is due to the fact that $RAT(r.a.a') = RAT(r.a), A(a; r.a : a')$ where $RAT(r.a)$ is given to AS a and $A(a; r.a : a')$ is an attestation by a itself. Moreover, since AS padding is allowed in BGP, a can form valid RATs for the form $RAT(r.a^i.a')$ for any neighboring AS a' and any $i \geq 1$, where a^i is a repeated i times. Let us call these

extensions of a RAT *transit extensions*.

Below we will define all possible transit extensions of a given set of routes. Then we will show that if the adversary can compute a valid RAT for a route that is neither in the set of routes for which it has seen a valid RAT, nor in the set of its transit extensions for those routes, then the adversary must have computed an existential forgery of an attestation.

Let $\mathcal{P}$ be a set of AS paths. Since all "good" routers check whether a path is almost simple, assume without loss of generality that all the paths in $\mathcal{P}$ are almost simple. Denote the transit extensions of $\mathcal{P}$ by x as $TE(\mathcal{P}, x)$. We define it iteratively as follows. First, for each $p \in \mathcal{P}$ all of the prefixes of p are added to $TE(\mathcal{P}, x)$, including p itself. Now, for each $p \in TE(\mathcal{P}, x)$, except for those that contain x, add the set $p.\{x\}^*$ and the set $p.\{x\}^+.\mathcal{Q}_{p,x}$ to $TE(\mathcal{P}, x)$, where $\{x\}^* = \{x^i \mid i \geq 0\}$ and $\{x\}^+ = \{x^i \mid i \geq 1\}$. Here $\mathcal{Q}_{x,p}$ is $\mathcal{ASN}$ minus x and minus the ASes in p and is defined so that all of the extensions are almost simple paths. As an example, if $(a, b, x, x, d,)$ is in $\mathcal{P}$, then we first add $(a), (a, b), (a, b, x), (a, b, x, x), (a, b, x, x, d)$ to $TE(\mathcal{P}, x)$. Next we add the following paths to $TE(\mathcal{P}, x)$

$$
\begin{array}{llll}
(a), & (a, x), & (a, x, x) \ldots, & (a, b), \quad (a, b, x), \quad (a, b, x, x) \ldots, \\
 & (a, x, b), & (a, x, x, b) \ldots, & \quad (a, b, x, c), \quad (a, b, x, x, c) \ldots, \\
 & (a, x, c), & (a, x, x, c) \ldots, & \quad (a, b, x, d), \quad (a, b, x, x, d) \ldots, \\
 & \vdots & \vdots & \quad \vdots \qquad \vdots
\end{array}
$$

Note that as defined $TE(\mathcal{P}, x)$ contains $\mathcal{P}$.

Definition 6. A secure RAT is defined as follows. An adversarial AS x is given access to a RAT oracle. That is, x can query the RAT oracle on routes r of its choice in a dynamic fashion and receive $RAT(r)$ for each of its queries. Let $\mathcal{P}$ be the set of such routes. A RAT forgery by x is a valid $(r, RAT(r))$ pair for some r not in $TE(\mathcal{P}, x)$, the set of transit extensions of $\mathcal{P}$. A RAT is secure if no time bounded adversary with access to a RAT oracle can compute a RAT forgery except with negligible probability. This definition of security can be parameterized in the standard fashion by a time bound, a query bound, and a probability bound but we omit the details of this parameterization here. These definitions lead to the main security lemma for RATs.

Lemma 1. If x has a strategy for efficiently computing a RAT forgery then there is an efficient strategy for computing an attestation forgery.

Proof. Let $\mathcal{P}$ be the set of paths for which the adversary x has received valid RATs. By definition, from this set of RATs, x can compute a valid RAT for each path in the transit extension of $\mathcal{P}$, $TE(\mathcal{P}, x)$. Let $(r, RAT(x))$ be x's RAT forgery. That is, $(r, RAT(x))$ is a valid (route, RAT) pair but r is not in $TE(\mathcal{P}, x)$.

Let s be the route in $TE(\mathcal{P}, x)$ that has the longest matching prefix with r of all of the routes in $TE(\mathcal{P}, x)$ and let p be that matching prefix path. Note that the construction of $TE(\mathcal{P}, x)$ is such that all of the prefixes of s are in $TE(\mathcal{P}, x)$ and thus r cannot be a prefix of s. Thus, either $s = p$ is a prefix or r, in which case $|s| = |p| < |r|$, or $r^{\leq}_{|p|} = s^{\leq}_{|p|} = p$ and $s_{|p|+1} \neq r_{|p|+1}$. By definition, $p_{|p|} = r_{|p|}$ is the last element of the prefix.

Note that adversary can extract easily the RAT for $r^{\leq}_{|p|+1}$ from the RAT for r just via the recursive definition. The last attestation in $RAT(r^{\leq}_{|p|+1})$ is simply $A(r_{|p|}; r^{\leq}_{|p|} : r_{|p|+1})$.

By definition of p and the fact that $|r| > |p|$, $r^{\leq}_{|p|+1}$ is not in $TE(\mathcal{P}, x)$. Hence, $A(r_{|p|}; r^{\leq}_{|p|} : r_{|p|+1})$ is not in any $RAT(q)$ for $q \in TE(\mathcal{P}, x)$. Thus $A(r_{|p|}; r^{\leq}_{|p|} : r_{|p|+1})$ is an attestation forgery under the condition that $r_{|p|} \neq x$. (Clearly, if $r_{|p|} = x$, x can legitimately compute $A(x; r^{\leq}_{|p|} : r_{|p|+1})$.)

We now argue that $p_{|p|} = r_{|p|} \neq x$. To see this, suppose that in fact $p_{|p|} = x$. Then all extensions of p by one AS a such that $p.a$ is almost simple are also in $TE(\mathcal{P}, x)$. But in such a case $p.a$ must be a prefix of r for some a, contradicting the maximality of p. □

The implication of the lemma is that if the attestations are secure as per the definitions above, then the route attestation tag will be secure as per the definition above. The security lemma and proof can be easily modified to include security parameters. That is, the above lemma can be extended to give the security parameters of the RAT scheme as a function of the security parameters of the underlying attestation scheme.

Note that the adversarial model and proof of security are quite strong. They allow an adversarial AS x to see RATs for any routes of its choosing including, for example, routes that do not correspond to actual topology and routes that may have already transited x and continued for several more hops. If x is colluding with another AS, it may indeed be able to see the latter RATs. The security model protects against forgeries even with this type of collusion. It is better, of course, to overestimate the power of an adversary since it is always difficult to bound the information that a determined adversary can uncover. Even with this more powerful model, the security of RATs reduces to the security of the underlying attestations.

It is possible to capture the case of several ASes colluding with definitions and a security lemma similar

to that above. However, the definitions are more complex and are omitted here. But, intuitively suppose a set of adversaries X is given valid RATs for a set of paths $\mathcal{P}$ of their choosing. The transit extensions of $\mathcal{P}$, $TE(\mathcal{P}, X)$, consist of all of the almost simple paths for which the adversaries can derive valid RATs from the valid RATs for the paths in $\mathcal{P}$. If the adversaries X can succeed in computing a valid (route,RAT) pair for a route not in $TE(\mathcal{P}, X)$ then they have succeeded in computing a RAT forgery. As long as it is computationally difficult to forge an attestation with non-negligible probability, it is computationally difficult for the adversaries X to compute a RAT forgery with non-negligible probability.

3.6 Replay Attacks

The security discussion above does not take into account replay attacks. One approach to prevent replay is for the sender to concatenate a strictly increasing value into the messages that it is authenticating and for the verifier to keep the largest value seen thus far as state (and discard items that have value less than or equal to it). Unfortunately, BGP announcements currently do not provide enough information to make this approach work. For example, different routers in an AS can choose and announce different best paths for the same prefix. Thus an upstream router may hear different paths for a prefix with the same AS in the AS path. Thus, even if time stamps are securely bound to each AS in an AS path, there is no guarantee of monotonicity of the time stamps even under normal operation.

The approach proposed by S-BGP [17] is for a router to include an expiration time in each of its attestations. When the expiration time is passed, the attestation is no longer valid. Since a RAT is only valid if all of its attestations are valid, a RAT times out whenever any one of the attestations in the RAT times out. If a (route, RAT) pair is about to expire due to the expiration time of a given router, that router can re-announce that route before expiration with a new expiration time. While this does not completely defeat replay attacks, it does limit them to a well-defined vulnerability window. However, there is an operational tradeoff with security. If the vulnerability window is too small, the number of route announcements may become excessive. A validity time that achieves a reasonable tradeoff is likely to be on the order of several hours, but a detailed engineering analysis is required to set this time properly. We assume a modicum of loose time synchronization among routers, and parameterization can help to determine the optimal window size for minimizing replay attacks while allowing for clock skew in routers issuing RAT attestations. Our attestations and RATs will follow the S-BGP approach.

A small change must be made to our prior set-up to allow for timestamps to be included in attestations.

Definition 7. Consider $A(a_i; r_i : a_{i+1}|c_i)$ as an attestation by a_i of the string $(r_i : a_{i+1}|c_i)$ where c_i is considered *ancillary information*. Of course, to verify this attestation, the verifier needs a_i's public key and the string $(r_i : a_{i+1}|c_i)$. Thus, we modify the recursive definition of a RAT slightly as follows:

$$RAT(r_i^{\leq}) = RAT(r_{i-1}^{\leq}), A(p_{i-1}; r_i^{\leq} : p_i|c_{i-1}), c_{i-1}$$

Using this definition, it is easy to see that, along with the route r itself, $RAT(r)$ contains sufficient ancillary information to verify all of the attestations within the RAT.

3.7 Topology

As discussed above, although topology is not taken into account in the definition of a secure RAT, topology does enter the picture in the fourth requirement for a valid (route, RAT) pair; that is, the pair must have been received over an authenticated eBGP session with a router in AS $a*$ where $a*$ is the last AS in the path. The requirement as stated only verifies a local topology condition, i.e., an eBGP neighbor relationship. The intent of the requirement is to achieve routes that are topology respecting, as defined above. If every "good" router obeys the four requirements the intent is nearly satisfied. The only deviation from topology respecting routes that can be achieved is by a coalition of adversaries who may create arbitrary almost-simple subpaths amongst themselves. Even in this case, however, when the route emerges from the adversarial coalition, each good router that hears the announcement will enforce the veracity of the edge from the coalition router and all subsequent edges will be topology respecting.

It is possible to strengthen requirement 4 of Definition 5 in a manner similar to soBGP[48]. Using an in-band or out-of-band mechanism, routers can send signed statements attesting to their AS number and those of their eBGP neighbors. In this manner, each router can maintain a relevant view of the BGP graph. Requirement 4' is then simply that r is topology respecting with the respect to the certified eBGP edges in the router. Of course, the coalition of adversaries can still create arbitrary almost-simple, topology-respecting subpaths amongst themselves. The main difference between 4 and 4' is that effectively, the coalition must publicly commit to their subgraph ahead of time. Whether this commitment sufficiently narrows the vulnerability enough to warrant the dissemination of certified eBGP edges is a question for further study.

3.8 Constructions

In this section, we describe the attestation schemes used by S-BGP and Nicol et al. In addition, we propose a different attestation scheme and several variants. S-BGP attestations include a validity interval $I = [t_s, t_e)$ and are denoted $A(a_1; r : a_2|I)$ As noted before, the S-BGP attestations are implemented as a public key signature. That is, $A(a_1; r : a_2|I) = I, \sigma$ where σ is the signature of the string $r : a_2|I$ using the private key of an AS a_1.

Both the Nicol et al. scheme and our schemes are based on set-membership proofs. Nicol et al. make crucial use of the fact that BGP-speaking routers do not continuously send BGP updates to their neighbors. Instead, BGP-speaking routers group route updates into 30-second intervals, and only send these updates to their neighbors at the end of the 30-second interval. For a given router in AS a, define $\mathcal{R}_t$ to be the set of all tuples $(r : a')$ such that route r is sent by the given router in an eBGP update to a router in a' in the 30-second interval ending at time t. The routers in this scheme create set-membership proofs for the set $\mathcal{R}_t$ for each time t that ends an interval. As discussed previously, ancillary information can be included in a set-membership proof. In this case, both the time of the update t and the expiration time of the announcements t_e are included. Let π_α be the set-membership proof for $\alpha \in \mathcal{R}_t$. Then for each $\alpha \in \mathcal{R}_t$ the attestation $A(a; \alpha|[t, t_e))$ is simply $(\pi_\alpha, [t, t_e))$. Since the router is sending the attestations for each $\alpha \in \mathcal{R}_t$, this set of attestations can be more parsimonious than the collection of individual attestations (we omit details for brevity). Nonetheless, as (route, attestation) pairs for routes represented in $\mathcal{R}_t$ are forwarded downstream in the appropriate RAT for an extension of the route, the amortized length of the encoding of (π_α) will increase. This is because fewer and fewer of the attestations for elements of $\mathcal{R}_t$ will be included in downstream updates.

Our scheme is similar to Nicol et al. in that we also use set-membership proofs. However, the method with which we choose to aggregate updates into sets is different. Assume for now that a router in AS a knows in advance all of the routes it will send during time interval $I = [t_s, t_e]$. That is, let $\mathcal{T}_I$ be the set of all tuples $(r : a')$ where the route r was sent by the given router to a router in a' within the interval I. Within the interval I, when the router needs to compute the attestation $A(a; \beta|I)$, for $\beta \in \mathcal{T}_I$, it computes the set-membership proof π from β, $\mathcal{T}_I$, and ancillary information I. The attestation for β is (π, I). Of course, a router cannot know in advance all of the routes it will receive in an interval I. However, as we will show in subsequent sections, for BGP updates, the past is a fairly accurate predictor of the future. Thus the

set $\mathcal{T}_I$ is an approximation based on past history of the routes that will be needed for updates in period I. When the router needs to send an attestation for a route not in $\mathcal{T}_I$, it simply computes an S-BGP attestation. If $\mathcal{T}_I$ is required to have a maximum size bound, as it must, then there are a variety of caching strategies for maintaining $\mathcal{T}_I$ from one interval to the next.

As we will see, for reasonably sized intervals I, the set $\mathcal{T}_I$ can get quite big. However, $\mathcal{T}_I$ can be partitioned into smaller sets in a number of ways, and then a set-membership proof scheme can be applied to each set. This affords a time-space tradeoff. For example, for all address blocks b in tuples in $\mathcal{T}_I$, let $\mathcal{T}_{b,I}$ be the tuples that have address block b. In what we denote the *prefix scheme*, the router creates set tags and set-tag signatures for each $\mathcal{T}_{b,I}$. And the attestations are membership proofs for the appropriate set and member of that set. In another variant, $\mathcal{T}_I$ is partitioned according to the origin AS. That is, we define $\mathcal{T}_{a,I}$ as the elements of $\mathcal{T}_I$ that share the same origin AS. The scheme based on this partition is denoted the *origin AS scheme*.

A final variant of our scheme allows the expiration time of an attestation to be different from the expiration time of the set-tag signature. For example, suppose we partition I into k subintervals. Let the set of intervals be $\mathcal{K}$. Using the origin AS scheme as an example, for each $\mathcal{T}_{a,I}$, the router creates a proof system for the set $\mathcal{T}_{a,I} \times \mathcal{K}$. Note that since the size of membership proofs can be made to be logarithmic in the size of sets, this only adds $\log |\mathcal{K}|$ to the length of the membership proofs. In this case, the output of an attestation is the same as above plus the particular subinterval used. That is I is used in the public key signature validation of σ and the subinterval is used to encode the leaf that the verifier must use.

Note that for every address block includes an empty path in $\mathcal{T}_I$. Within our setting, we consider a withdrawal of an address block to be denoted by a route advertisement of that address block with an empty path.

4 Path Stability

This section analyzes the central hypothesis upon which our cryptographic constructions are based: the set of paths for a prefix or emitted from an AS are small and stable over time, i.e., ASes exhibit path reference locality. The following experiments evaluate path *density* (number of distinct paths observed from peers and other points across the Internet) and *stability* (rate of discovery of new paths). In these experiments, we examine data from the 40 listening points of the Route Views [25] BGP repository. Each listening point

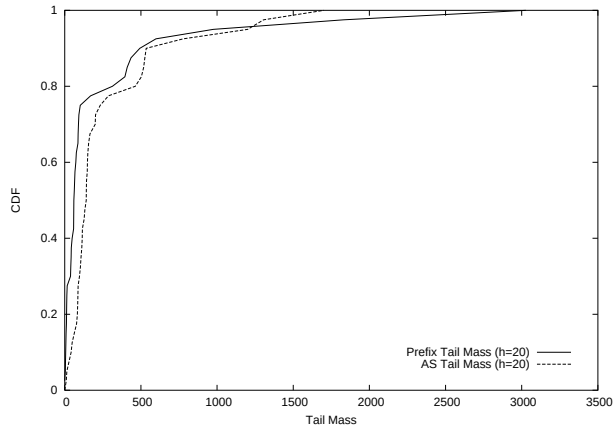

Figure 3: Tail mass - CDF of tail mass for 40 Route Views listening points during February 2004.

Tail Mass Test	Min (LP)	Median (LP)	Max (LP)
Prefix (h=10)	67 (#23)	1,178 (#8)	17,784 (#40)
Prefix (h=20)	0 (#23)	63 (#17)	3,027 (#40)
AS (h=10)	163 (#23)	1,135 (#8)	4,967 (#40)
AS (h=20)	10 (#23)	142 (#8)	1,701 (#40)

Table 1: Listening point tail mass

dataset represents a transcript of all UPDATE messages received by that monitored AS. point).

We are not the first to characterize path stability. Other studies use the available BGP data to investigate the number of unique paths to a prefix assuming connectivity to two listening points over a single day [9], to estimate the number of cryptographic operations required for prefix validation [46], to establish a delegation hierarchy [21], and to examine address allocation and routing table growth [3], scalability of router memories [29] and table fragmentation [22], or to ascertain the stability of popular routes [40]. We found these past analysis instructive but incomplete for our purposes. These analyses focused on instantaneous table size or growth over time, or considered only a small subset of prefixes. The current work required a characterization of total unique paths an observer sees per-AS and per-prefix on a continuing basis. Hence, while past studies largely focus on growth trends, our analysis required a finer characterization of path *churn*. Detailed below, these requirements prompted the study of AS/prefix *tail mass* and path *rates of discovery*.

We begin our analysis by using *tail mass* to measure path stability. Tail mass $T_h(k)$ is the number of unique values above a threshold h encountered by observer k. This study is concerned with number of unique paths, so we calculate tail mass as the number of prefixes or ASes that have more than h unique path vectors associated with them. Intuitively, tail mass shows how many prefixes or ASes have a "large" number

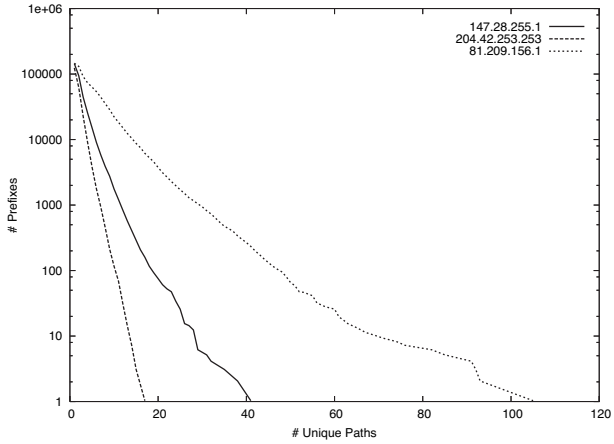

Figure 4: CCDFs of unique paths per prefix measured from multiple Route Views listening points, for February 2004.

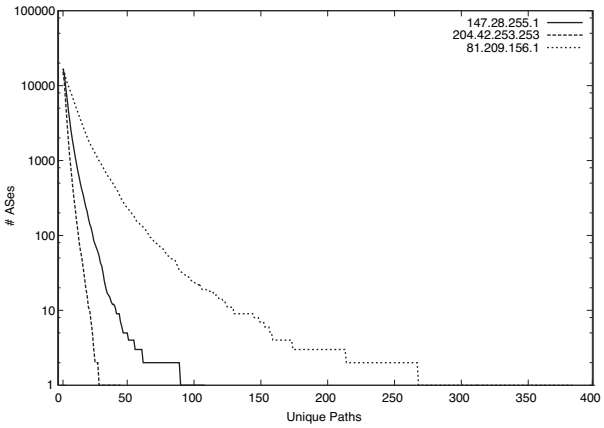

Figure 5: CCDFs of unique paths per AS measured from multiple Route Views listening points, for February 2004.

of paths associated with them (as defined by a threshold h). The following is based on the analysis of the 217,707,968 updates observed by the 40 listening points during February 2004.

Figure 3 shows a cumulative distribution function of the prefix and AS tail masses of each listening point when the threshold is 20 (h=20). A striking aspect of this data is its density, where 80% of the listening points have a tail mass less than 500, and 67% have masses less than 200. This indicates significant stability at the listening points.

Table 1 summarizes the most, least, and median-stable listening points as represented by tail mass, measured across several experiments. The data suggests candidate *representative* listening points as models for minimum, maximum, and typical stability. As such, we select listening point 23 (204.42.253.253) as maximally stable (i.e., has the smallest tail mass), point 40 (81.209.156.1) as minimally stable, and point 8 (147.28.255.1) as typical in the following experiments.

We now use the representative listening points to more closely scrutinize path stability. Figure 4 shows a CCDF for the unique number of paths observed by the listening point associated with various prefixes. In the average case, less than 2% of prefixes have more than 10 paths associated with them, and less than 0.06% more than 20. In the worst case, 15.3% of prefixes have more than 10 unique paths, 2.57% have more than 20, and 1.17% have more than 25.

Figure 5 shows a CCDF for the observation of unique paths by AS. Because the number of ASes a listener sees is little more than 10% of the total number of prefixes seen, we would expect that the number of unique paths per AS would be correspondingly larger than in the per-prefix case. However, the difference

is not as pronounced because many prefixes originating from the same AS will have the same path. This vector will be counted n times for n different prefixes, but only once if they all originate from the same AS. For the average case, we found that 6.90% of ASes have more than 10 unique paths for at last one prefix in the AS, and only 1.00% have more than 20 unique paths. In our worst case, 33.2% of ASes have more than 10 unique paths, 11.1% have more than 20, and 5.17% have more than 30.

The path lengths for the minimally stable listener (81.209.156.1), as illustrated in Figures 4 and 5, are considerably longer than for other listeners. Doing a WHOIS lookup against the RIPE database shows that the block 81.209.156.0/24 is used for loopbacks and point-to-point links within the USA, and is owned by LambdaNet Communications Deutschland AG, of Hannover, Germany. A traceroute to the IP address shows the router labeled `dc-1.us.lambdanet.net`, implying that it is located in or around Washington, DC (the next-to-last hop is labeled `ashburn`, likely referring to Ashburn, VA, making this assumption plausible). It is not clear, however, why the router would see so many more unique paths. We posit the route filtering of it and its peers followed a different policy from most other routers, or perhaps there are other factors at work. More study of this listener and its AS home (13237) may yield answers as to its unusual behavior.

A final series of tests assess the stability of the set of observed paths. Centrally, these tests attempted to estimate listening point *rates of discovery*. The experiments compute the frequency with which new paths are observed. We classify newness with respect to the AS (new when the AS has never advertised the particular path before) and prefix (the prefix has never been advertised with the path). Using the previously defined listening points, we examine the period between January 2003 and March 2004; the rates of discovery are shown in figures 6 and 7.

Two trends emerge from this study. First, there is nearly an order of magnitude difference between the number of new paths discovered per AS versus per prefix. An AS can have many different prefixes, each advertising the same AS path. Hence, when classification is done by origin AS, the path is only counted once, versus n times for n different origin prefixes. Although difficult to observe in the figures, a second trend shows strong discovery periodicity. We found that regular periods of little discovery corresponded to weekends. The network is at its most stable on the weekend, and hence little activity was observable in the BGP feeds.

The preceding results support our intuition that the set of known paths are not only stable over time, but the amount of churn between known paths is relatively small. That is, the paths observed in operational

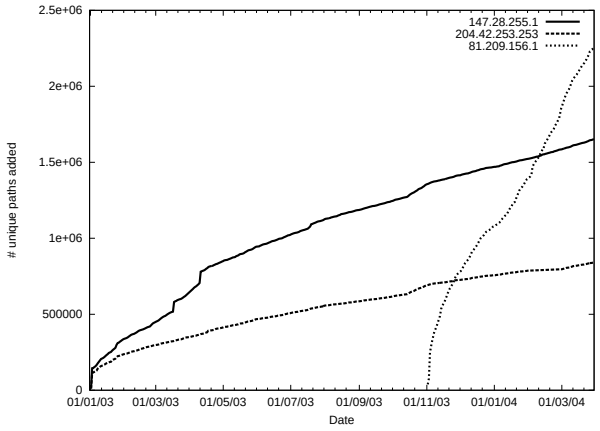

Figure 6: Rate of discovery CDFs for new paths per prefix as seen by multiple Route Views listening points, Jan 2003 to Mar 2004.

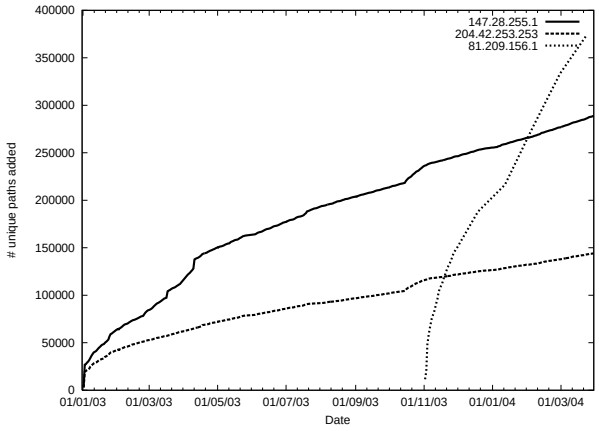

Figure 7: Rate of discovery CDFs for new paths per AS as seen by multiple Route Views listening points, Jan 2003 to Mar 2004.

environments are enormously dense and stable. Hence, there is an opportunity to exploit the reference locality. We explore how our constructions use this fact to implement efficient security in the next section.

5 Evaluation

In this section, we evaluate the efficiency of the constructions defined in the proceeding sections via trace-based simulation. We compare our solutions against S-BGP and its variants, and draw general conclusions about the effectiveness of the proposed optimizations.

5.1 Experimental Setup

Developed specifically for this work, the *pasim* simulator models a single AS on the Internet and measures the computational and bandwidth costs associated with the validation of received paths. Computation is measured by the number of signature validations, which dominate all other computational costs (e.g., buffer handling, etc.) making them a good cost approximation. The simulations measure the amount of bandwidth consumed by the received proofs, but do not consider bandwidth consumed by other non-security related bandwidth costs (e.g., control traffic). We do not simulate the costs associated with the generation of proofs. Because structures are signed with low frequency (days), these costs will be dominated by validation. The simulations reported in this section use BGP update data collected during January 2004. Based on the results

from the previous section, we ran simulations for the "typical" listening point (`147.28.255.1`).[5]

We simulate S-BGP route attestations and the signature amortization scheme proposed by Nicol et al. [31], which groups route updates into intervals and sends when the 30-second BGP timer is triggered; these updates are signed over a Merkle hash tree [6]. We contrast these schemes with simulations of our constructions: the prefix scheme, origin AS scheme, and the all AS paths scheme as defined in the preceding sections.[7] Each timed UPDATE in the trace data is played back to the simulated BGP router and processed according to the simulation solution. Unless described otherwise, all tests in this section assume that received signatures are hashed and kept in a 16 MB cache (described in further detail below), with simulated tree-based proof systems regenerated every 24 hours and authentication proofs issued every hour, absent changes requiring new attestations or signatures to be generated (e.g., a change in the best path to a destination, or the addition of one or more new paths to the path membership set).

The simulation of our tree-based proposed schemes requires knowledge of all the paths advertised by an AS, which cannot be determined from a single listening point (or even by the entire 40 that comprise the Route Views corpus of data). One observation we make is that we are likely to see more unique paths from those ASes we are closest to. We approximate this by assuming unique paths comprise 7/8 of the paths observed from those ASes one hop away, 6/8 from ASes two hops away, etc., and adjust the tree size appropriately.[8] More precisely, if u unique paths associated with a proof system for an AS h hops away are seen, the proof system size is approximated to be $u(2 - h/8)$, e.g., $h = 3, u = 16 \rightarrow s = 16(13/8) = 26$. Note that an over or under estimate will affect the simulated size of the proofs, but will not impact the amount of computational resources needed to validate them.

5.2 Simulation Results

Our initial simulations compare computation and bandwidth usage. Figure 8 shows the number of signatures used by each scheme. S-BGP consumes the most computational resources validating signatures. The Nicol optimization effectively reduces these costs by half. This drop is due to the amortization of signatures

[5] We repeated the tests in the most and least stable listening points. In all cases, the costs scaled with the number of unique paths and rates of discovery as discussed in the preceding section.

[6] As previously mentioned, we do not model the aggregate signatures introduced in [52] as these optimizations are orthogonal to our main goal in comparing constructions; such optimizations are considered for future work.

[7] We simulated operation of the final variant of our scheme described in section 3, where expiration time of the attestation could be different from expiration time of the set-tag signature. The results differed from our origin AS scheme by a small factor. Hence, for clarity we omit these results from the graphs.

[8] We conservatively chose 8, as we observed that paths of four or more hops from the core were typically originated by stub ASes.

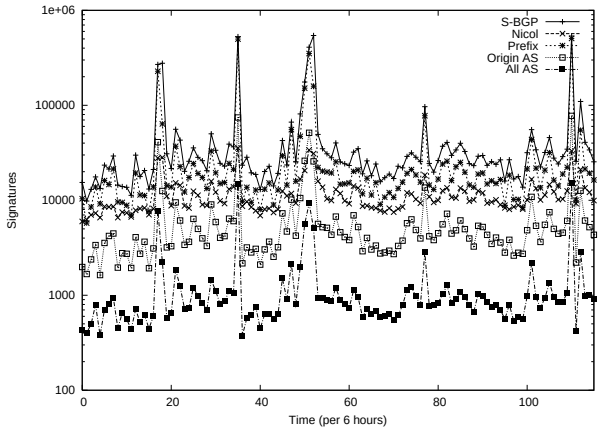

Figure 8: Validation cost - signatures validated per hour for *S-BGP*, the *Nicol et al.* scheme, and our constructions: *prefix path*, *origin paths*, and *all path* validation.

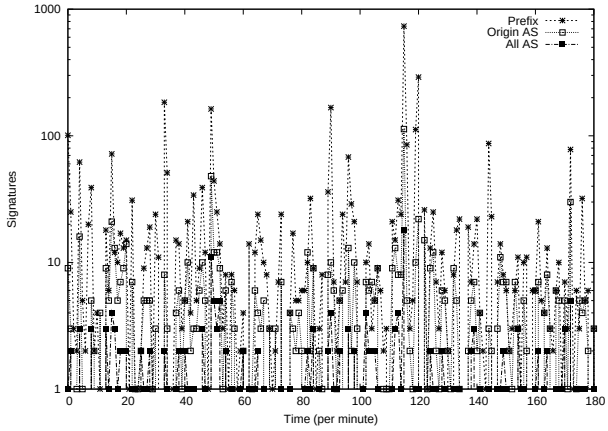

Figure 9: Instantaneous signature validation - per one-minute validations in simulated *prefix path*, *origin paths*, and *all path* validation.

across the 30-second time period. Interestingly, this indicates that, on average, only a few paths propagate through an AS in a given time period. Because of the sustained load, the data lets us posit that optimizations over short periods (such as Nicol et al.) are likely to be less effective than longer periods, even if the latter may require more resources (e.g., large structures, more retained state). The tree-based solutions require fewer validations than S-BGP. The prefix solution reduces the load by about 1/3. This is the effect of amortization over prefixes. Prefixes are largely stable and offer few paths, particularly over short time scales. Announcements for most prefixes will only be observed one or a few times per day. Hence, there is little opportunity to optimize. Note, however, that schemes such as SPV amortize costs in a fashion orthogonal to ours. Using our constructions in conjunction with those schemes could potentially reduce computational costs even further. The remaining AS path optimization schemes dominate all others: the origin paths

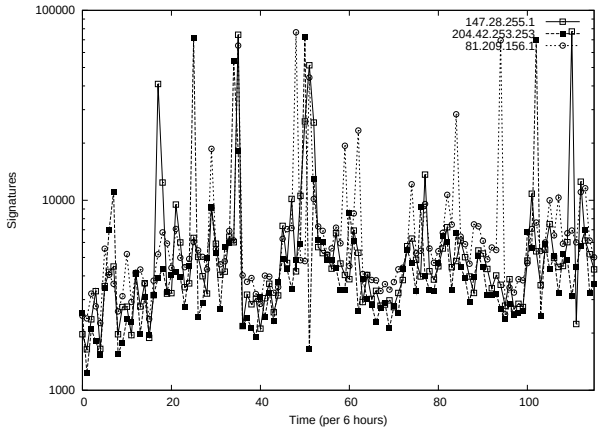

Figure 10: Signature validations by listening point - validations for the origin paths scheme per 6 hour period.

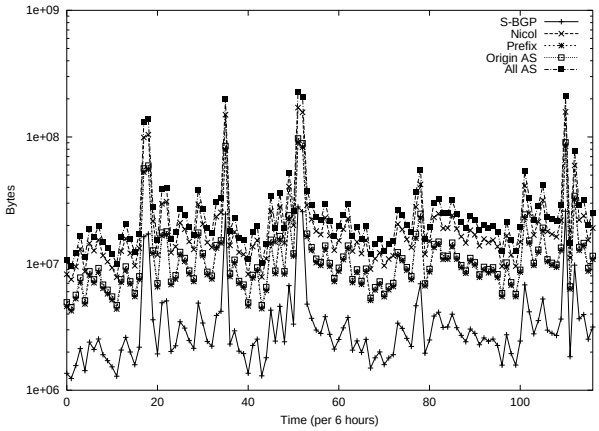

Figure 11: Bandwidth Cost - the number of bytes consumed by the transmission of the simulated path validation approaches.

scheme represents an 86.3% reduction, and the all paths a 97.3% reduction in signature validations over S-BGP. In a given 24 hour period, the maximum number of signatures encountered will be two times the number of active ASes (assuming that all path proofs expire at some point during the day, and are recreated). The origin paths are somewhat more costly because they fail to fully exploit the opportunity to amortize cost.

Hashing typically consumes vanishingly small amounts of computational resources compared to signature validation; it is approxmiately 1,000 times faster than RSA signature validation [9]. However, in some schemes, hashing can be performed frequently enough that it potentially impacts performance. For instance, we found in the *all-path* construction, because the tree was so large, the computational cost was equivalent to one and a half signature validations. However, in all other cases, hashing was dominated by the signature validation costs.

Figure 9 shows the *instantaneous rate* of signature validations, to model how many updates per minute are processed by the router for a three hour period at the beginning of January 2004 (the time-scale has been shortened to ensure readability of the graph). We found many bursts where many validations are necessary per minute, particularly in the prefix scheme (where on average a burst would require less than 30 signature validations, but rare peaks would require a hundred or more). The origin scheme, which strikes the best compromise between validations and bandwidth, generally requires under 10 validations per minute, or one every six seconds on average.

Figure 10 shows the number of validations required for the origin scheme at the three listening points. The listening point demonstrating worst-case behavior has a number of bursty points with significant num-

bers of validations required; however, this burstiness is evident in all schemes and is constant across listening points.

Demonstrated in Figure 11, the bandwidth costs are largely the inverse of signature costs. S-BGP consumed far less bandwidth than the other approaches, because it generates small proofs. The prefix and Origin AS approaches were significantly more costly, consuming 3.35 and 3.57 times more resources than S-BGP, respectively. Interestingly, Nicol was second only to the all path scheme in consuming resources. The Nicol scheme creates a tree for every 30-second quantum, and subsequently sends a potentially large set of succinct proofs every period. The all path scheme was by far the most costly approach, consuming about 6 times as much bandwidth as S-BGP. In this case, the average bandwidth consumed per 6 hour period is 77 kilobytes. However, this approach may be prohibitive due to short bursts, which required as much as 139 megabytes in a single minute.

Any path authentication scheme must allocate storage resources for security relevant state (e.g., cryptographic proofs). In S-BGP, the additional space requirements to hold route attestations is estimated to be between 30 and 35 MB per BGP peer, though it is suggested that memory requirements in asymmetric peering relationships, such as between a large ISP router and a number of smaller peers, would be lower [14]. The storage requirements of the schemes proposed in this paper are unique to their design. Recall that the prefix approach requires every prefix to have a proof structure, while the all path approach requires a proof per AS; these two schemes form maximal and minimal requirements, respectively. Our simulations show that the total cost of storing *all proofs* across all peers ranges from approximately 55-60 MB for the prefix scheme to under 10 MB for the all path scheme. In the origin AS scheme, the total cost is approximately 25 MB.

The simulations illustrated in Figure 8 assume a proof cache of 16 MB. In our simulation model, this cache is separate from the storage space for the full set of proofs. We make this design decision so that the cache could be accessed more rapidly by the router as part of its fast path packet processing, but retain access to the proofs in stable storage (as needed for announcement creation). The additional stable storage costs are not onerous, and could likely be stored in memory itself on larger routers. Alternately, even smaller routers (e.g., Cisco 3600 series) include slots for flash memory, and are capable of accepting cards with 256 MB or greater, well above the requirements of our scheme. We assume that in real systems, to keep the cache size at a minimum, a hash of a received signature is stored in cache, rather than the signature itself. The router hashes the signature of an incoming update and checks whether it appears in the cache. If it is, a

signature validation is not necessary. Hashed signatures are expired from the cache on a LRU basis. When sending an update, the full proofs to be sent are retrieved from stable storage.

For simplicity, we assume a single-level cache, where if the signature hash is dropped from the cache, a revalidation is necessary. Multi-tiered caches, where the verified signature would be kept in another level of memory if it expired from the cache, would serve to decrease the cost of signature validations further. These and other optimizations related to router architecture are deferred for future work.

6 Discussion

6.1 Incremental Deployment

A major difficulty of retrofitting security is the need for *incremental deployment*. Simply put, there are large portions of the Internet that will adopt solutions slowly or not at all. Any feasible solution must be designed such that communities of interested parties can work collaboratively to provide a working, secure system. Moreover, functionality can not come at the expense of poorly equipped enterprises. Such approaches would disenfranchise people and networks, and reduce universality of the Internet. However, those who do not participate need not receive benefit from deployment.

Past systems such as IRV [5] addressed incremental deployment by performing security *out-of-band*. They allow parties to exchange data without any change to BGP. Those who wish to exchange security relevant data do so freely over any mechanism that is available and convenient. However, this approach only works when the network is otherwise healthy or alternate channels are available. psBGP takes another tack in which the parties police each other's activities [46]. The incremental deployment approach in psBGP is one of a mutual embrace: like soBGP, communities of peers must work in concert to achieve a larger security posture.

We adopt this latter scheme, where communities of like-minded organizations will organically form *unions* of ASes. These unions will mutually authenticate credentials to be used in the issuance of proofs of authentication, as formally discussed in section 3. At the protocol layer, we adopt a similar strategy to S-BGP of signing transitions to and from non-adopting ASes. Of course, knowing which ASes are participating in the protocol is essential for ascertaining the validity of received routes. In a sense, our approach is similar to the S-BGP protocol, and as such can make use of its procedures and structures for incremental deployment. S-BGP purposefully specifies that the route attestation be defined as a transitive discretionary (optional)

attribute in BGP-4 for both eBGP and iBGP updates [16]. In this way, routers that have not implemented route authentication can simply ignore the Route Attestation Tag but are nonetheless required to forward it to the upstream routers. We address here the security implications of incremental deployment.

ASes that wish to deploy path authentication must generate public/private key pairs for use by their routers in computing attestations and be granted bindings (certificates) between their AS number and public key values by the relevant PKI [42]. Any AS can, of course, verify attestations without themselves having their own certified public/private key pairs. They need only know the public keys and certificates of the current "certified attesters." Given that the primary security benefit derives from verification and the primary cost derives from computing attestations, we assume that every certified attester is also a verifier.

We assume for now that every certified attester knows the AS number of every other certified attester. The definition of a RAT must be modified slightly as follows. For $r = (b; p_1, \ldots, p_{|p|})$

$$RAT(r_i^{\leq}) = \begin{cases} RAT(r_{i-1}^{\leq}), A(p_{i-1}; r_i^{\leq} : p_i) & \text{when } p_{i-1} \text{ is a certified attester, and} \\ RAT(r_i^{\leq}) & \text{otherwise.} \end{cases}$$

As an example, let $p = (a_1, a_2, a_3, a_4)$, where a_1 and a_3 are certified attesters. Then

$$RAT(b; a_1, a_2, a_3, a_4) = OAT(b, a_1), A(a_1; (b; a_1) : a_2), A(a_3; (b; a_1, a_2, a_3) : a_4).$$

Note that a_3's attestation is over the entire path, not just the subset of the paths that are certified attesters. This issue of incremental deployment of origin authentication tags was discussed extensively in [21].

Of course, a subpath of the actual AS path traversed by a route that goes through ASes that are not certified attesters can be spoofed to appear to be a different subpath. However, that subpath is constrained to be an almost simple subpath consisting only of ASes that are uncertified attesters. In addition, verifiers can apply other topological constraints of which they are aware. For example, we assume that certified attesters use authentication on their eBGP session and can thus validate the AS number of the AS sending the update over that BGP session. Hence, in the example above, although a_2 is not a certified attester, given the structure of the RAT and the topological constraint imposed by a_3, a_2 cannot insert an almost simple subpath into the route. In the above setup, as the number of certified attesters increases, the set of spoofable subpaths decreases and the security incrementally increases.

We return to the assumption that all certified attesters know the AS numbers of the other certified at-

testers. This is an important requirement. Without it, at any time, a certified attester may simply pretend to be uncertified and start spoofing subpaths. In such a case, there can be no guarantee of incremental gain in security as the number of certified attesters grow. At a high level, it is natural for the PKI to distribute the public key certificate of a newly certified attester to all of the existing certified attesters. After all, this certificate is needed by the existing attesters to verify the attestation of the newly certified attester. Thus, the requirement is completely natural. Note that it introduces a slight modification to the verification procedure for validating a RAT, as all AS numbers of certified attesters in a route must be accompanied by the appropriate attestation; otherwise, the RAT is invalidated. The issues concerning the mechanisms and tradeoffs for distributing the certificates is extremely important but beyond the scope of this paper.

6.2 Worm-holing

Preventing *Worm-holing* is enormously difficult. There is nothing preventing an AS from achieving an arbitrary connectivity, and as such there is little one can do within a security protocol. Protocols such as soBGP do an approximate job of prevention by authenticating the network structure in the topology database. This prevents transient AS compromise from affecting the system as a whole, but does nothing against the truly adversarial AS. We argue that the real solutions to worm-hole prevention lie in good network management. For example, a large ISP should, and often does filter multi-hop advertisements from stub ASes (ASes with no other connectivity other than that provided by the ISP). Taken more generally, experience and formal relationships between networks are accurate sources of information for what constitutes good and bad connectivity.

6.3 Validation Optimizations

Kent et al. [15] have suggested a path validation optimization aimed at reducing the load on validating S-BGP speaking routers. This optimization dictates that paths are validated only when they are selected as the *best paths*. However, it is not clear the degree to which this optimization will mitigate the computational costs of S-BGP. Consider an AS A with k neighbors. Any prefix p will be reachable through j neighbors, where $0 \leq j \leq k$, and j routes will be held by the AS. The fractional computational savings f for a given prefix on a given router over a period of time Δ is just the ratio of updates sent for that prefix during Δ divided by the total number of updates received for that prefix during Δ. Of course, f will vary from router

to router and prefix to prefix, but f is likely to be on the order of $1/j$ for j defined above. For the data collected in our study, the median number of unique paths per prefix was 2.5 and the mean value was 2.8. A careful study of f remains for future work. But we note here that the same optimization can be used for our authentication proofs based on set-membership proofs. We will also achieve a factor f computational speedup. That is, when the optimization is applied to both schemes, the ratio of the computational overheads will remain the same.

7 Related Work

Interdomain routing security has been studied for some time [34, 43], but comprehensive and efficient solutions remain elusive. The following considers how several of these efforts address path security.

Possibly the most comprehensive solution advanced to date, the Secure Border Gateway Protocol (S-BGP) [17, 16, 42] uses a public key infrastructure to support the authentication of routing artifacts. The S-BGP PKI maintains certificates for each AS and S-BGP-speaking router. Every router includes a *route attestation* with each advertisement. The route attestation is a signed statement of the AS identity, the paths, the prefix and the AS to which the announcement is directed. The S-BGP speaker also includes the route attestation of the route on which the advertisement is based. This prevents an adversary from adding or removing ASes from the path. While the authors of S-BGP have introduced a number of optimizations that reduce resource consumption [15], the costs associated with it are viewed as limiting factor in many environments [5, 9, 46]. For example, Nicol et al. showed that, under a set of timing and cost assumptions, such costs can double the path convergence time [31]. However, Nicol et al. did not model optimizations reported in [15]. It is not clear if and how the optimization would affect convergence times. While some argue that co-processors and protocol optimizations may make computation feasible, storage remains a major problem. Kent estimates that S-BGP will require an additional 30-35 megabytes of storage per peer [14]. Such costs are manageable in routers with a few peers, but are problematic in large ISPs or exchanges. However, Kent further argues that there are asymmetric configurations where only a few routes are accepted (as in customer/ISP peering), and hence these situations would require fewer resources.

Partially in deference to the costs associated with more comprehensive solutions, the soBGP and IRV projects sought other means of addressing BGP security. The soBGP [30] protocol uses a topology database to validate that advertised paths are consistent with the signed statements of connectivity between ASes.

While this approach provides a limited security guarantee, it is effective in preventing a wide array of path hijacking and worm-holing attacks. However, soBGP does *not* provide path authentication, but simply implements a mechanism for detecting routes that are inconsistent with the authenticated topology. Philosophically similar to the earlier routing registry projects [23], the Interdomain Routing Validation (IRV) [5] project was motivated by the observation that any solution requiring a change to BGP was likely to be adopted slowly, if at all. IRV servers use an out-of-band (e.g., external to BGP sessions) protocol to exchange validation information. IRV is reliant on the routing infrastructure to extract and exchange routing data. Hence, unless some other infrastructure is put in place (e.g., static routes), the system is unable to function when connectivity is not available.

Validation of prefix ownership is essential to secure BGP. If not provided, an adversary can *hijack* entire networks by simply advertising the prefixes associated with them. Originally studied by Kent et al. [17, 42], an origin authentication (OA) service validates that an AS has the right to be the origin of a prefix. In a later work, Aiello et al. extended the study of OA by considering the semantics and efficient cryptographic constructions of origin authentication [21]. Principally, they explored formal semantics of the use and delegation of the IP address space. The set of all delegations between ICANN [12], registries, and organizations is modeled as a delegation hierarchy. Recently, van Oorschot et al. suggested an alternative low-cost but weak form of origin authentication, in which all BGP neighbors police and attest to the validity of the prefixes that an AS originates [46]. However, this is limited, as colluding ASes can forge origin information.

Several proposals have sought efficient constructions for BGP security. Hu et al. introduced the concept of cumulative authentication for securing route advertisements in path vector protocols [8]. They use the TESLA timed key release authentication to validate announcements using low cost symmetric key cryptography. TESLA is limited in that it requires tight time bounds on message transmission, which is in conflict with protocols built on asynchronous propagation protocols such as BGP. More recently, Hu et al. introduced the Secure Path Vector Protocol (SPV) [9], which also seeks to implement BGP path security using low cost cryptography. SPV creates cascading authenticators over many (low cost) one time signature structures.

The Whisper protocol [44] uses a mechanism that detects inconsistencies in received routes using RSA-style [41] cryptographic operations. To simplify, any conflicts between routes received from multiple peers emanating from the same original advertisement is detectable. Another approach that does not rely on a PKI or any form of cryptography is Pretty Good BGP [13], which relies on the stability of pre-existing

routes as an indicator of their veracity. Longer-lived, more stable routes are preferred over newly appearing routes, which may require a secondary verification to determine if they are valid. Because of the lack of provable security, this solution is considered a stopgap measure to provide a modicum of protection until a cryptographic solution is implemented. Other solutions that provide an alert-based approach include PHAS [19], a prefix hijacking alert system that examines routing updates from Route Views and RIPE reposoties for changes in the origin and notifies the owner of a prefix who registers with the service if updates have changed. The system is incrementally deployable in that to join the system, a prex owner need only register with the PHAS server; however, this server is also a single point of failure in the system, and if it is compromised, it could send out numerous false alarms to prex owners. Hu and Mao [7] also examined prefix hijacking and develoepd mechanisms for detecting real-time attacks by fingerprinting networks and hosts. This approach relies on a real-time monitor for updates, which must be available during crticial periods. Placing these monitors for optimal route coverage was also considered [51].

8 Conclusions

In this paper we have explored a range of cryptographic optimizations for securing BGP paths. Centrally, we exploit the stability of path advertisements to amortize cryptographic operations over many validations. This stability is confirmed via empirical analysis: the number of paths used by a particular AS for a given prefix is both small and largely constant over time. Through trace-based simulation, we show that our constructions reduce the computational costs of path authentication by as much as 97% over existing approaches, and show that other storage and bandwidth costs are nominal.

The problems of BGP security are sufficiently important to warrant discussion in the United States National Strategy to Secure Cyberspace [33]. This work studies tradeoffs between computational, bandwidth and storage costs for a range of BGP security path authentication mechanisms and is a step in a larger communal effort to design and deploy BGP security. The ultimate goal is to develop a comprehensive understanding of the security, cost, and manageability tradeoffs for BGP, to inform sound engineering decisions for future deployments. To this end, we plan to extend our evaluations to a range of realistic network environments, and to study the integration of optimizations suggested by others.

Acknowledgements

We are grateful to Jennifer Rexford for her detailed comments and patient answers to the many questions we posed to her. In addition, we would like to thank Nick Feamster, Tim Griffin, and Zhouqing Morley Mao for their feedback, and members of the SIIS lab at Penn State for commenting on multiple iterations of the paper.

References

[1] M. Baltatu, A. Lioy, F. Maino, and D. Mazzocchi. Security issues in control, management and routing protocols. *Computer Networks (Amsterdam, Netherlands: 1999)*, 34(6):881–894, 2000. Elsevier Editions, Amsterdam.

[2] A. Barbir, S. Murphy, and Y. Yang. Generic Threats to Routing Protocols (*Draft*). Oct. 2006. RFC 4593.

[3] S. Bellovin, R. Bush, T. Griffin, and J. Rexford. Slowing routing table growth by filtering based on address allocation policies. http://www.research.att.com/jrex/, June 2001.

[4] K. Butler, T. Farley, P. McDaniel, and J. Rexford. A Survey of BGP Security Issues and Solutions. Technical Report TD-5UGJ33, AT&T Labs - Research, Florham Park, NJ, Feb. 2004. (*revised June 2004*).

[5] G. Goodell, W. Aiello, T. Griffin, J. Ioannidis, P. McDaniel, and A. Rubin. Working around BGP: An incremental approach to improving security and accuracy of interdomain routing. In *Proceedings of NDSS '03*, Feb. 2003.

[6] M. Goodrich, R. Tamassia, and A. Schwerin. Implementation of an authenticated dictionary with skip lists and commutative hashing. In *Proceedings of DARPA Information Survivability Conference and Exposition II (DISCEX)*, Los Angeles, CA, June 2001. IEEE Computer Society Press.

[7] X. Hu and Z. M. Mao. Accurate Real-time Identification of IP Prefix Hijacking. In *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, USA, May 2007.

[8] Y. Hu, A. Perrig, and D. Johnson. Efficient security mechanisms for routing protocols. In *Proceedings of NDSS '03*, Feb. 2003.

[9] Y.-C. Hu, A. Perrig, and M. Sirbu. SPV: Secure Path Vector Routing for Securing BGP. In *ACM SIGCOMM*. ACM, August 2004.

[10] G. Huston. BGP Reports, May 2005. `http://bgp.potaroo.net/`.

[11] IANA. Autonomous System Numbers, March 2003.

[12] ICANN. The Internet Corporation for Assigned Names and Numbers, July 2004. `http://www.icann.org/`.

[13] J. Karlin, S. Forrest, and J. Rexford. Pretty Good BGP: Improving BGP by Cautiously Adopting Routes. In *Proceedings of the 14th IEEE International Confernce on Network Protocols (ICNP)*, Santa Barbara, CA, USA, Nov. 2006.

[14] S. Kent. Securing the Border Gateway Protocol. *The Internet Protocol Journal*, 6(3), Sep. 2003.

[15] S. Kent. Securing the Border Gateway Protocol: A status update. In *Seventh IFIP TC-6 TC-11 Conference on Communications and Multimedia Security*, Oct. 2003.

[16] S. Kent, C. Lynn, J. Mikkelson, and K. Seo. Secure Border Gateway Protocol (S-BGP) Real World Performance and Deployment Issues. In *Proceedings of NDSS '00*, Feb. 2000.

[17] S. Kent, C. Lynn, and K. Seo. Secure Border Gateway Protocol (S-BGP). *IEEE Journal on Selected Areas in Communications*, 18(4), Apr. 2000.

[18] C. Kruegel, D. Mutz, W. Robertson, and F. Valeur. Topology-based detection of anomalous BGP messages. In *Proceedings of RAID '03*, Sept. 2003.

[19] M. Lad, D. Massey, D. Pei, Y. Wu, B. Zhang, and L. Zhang. PHAS: A Prefix Hijack Alert System. In *Proceedings of the 15th USENIX Security Symposium*, Vancouver, BC, Canada, Aug. 2006.

[20] L. Lamport. Password Authentication with Insecure Communication. *Commun. ACM*, 24(11):770–772, Nov. 1981.

[21] P. McDaniel, W. Aiello, K. Butler, and J. Ioannidis. Origin Authentication in Interdomain Routing. *Computer Networks*, 50(16):2953–2980, Nov. 2006.

[22] X. Meng, Z. Xu, L. Zhang, and S. Lu. An analysis of BGP routing table evolution. Technical Report TR030046, Computer Science Department, UCLA, Jan. 2003.

[23] Merit Network. The Internet Routing Registry, July 2004. http://www.irr.net/.

[24] R. Merkle. Protocols for public key cryptosystems. Oakland, CA, Apr. 1980. IEEE Symposium on Research in Security and Privacy.

[25] D. Meyer. The Route Views Project, Nov. 2006. http://www.routeviews.org/.

[26] D. Meyer and A. Partan. BGP Security, Availability,and Operator Needs. NANOG 28, June 2003.

[27] S. Murphy. BGP Security Vulnerabilities Analysis. RFC 4272, Jan. 2006.

[28] M. Naor and K. Nissim. Certificate revocation and certificate update. In *Proceedings of the 7th USENIX Security Symposium*, Jan. 1998.

[29] H. Narayan, R. Govindan, and G. Varghese. The impact of address allocation and routing on the structure and implementation of routing tables. In *Proceedings of ACM SIGCOMM '03*, Karlsruhe, Germany, Aug. 2003. ACM.

[30] J. Ng. Extensions to BGP to support Secure Origin BGP (soBGP), Oct. 2002. Internet Draft.

[31] D. Nicol, S. Smith, and M. Zhao. Evaluation of efficient security for BGP route announcements using parallel simulation. *Simulation Modelling Practice and Theory*, 12(3–4):187–216, July 2004.

[32] O. Nordström and C. Dovrolis. Beware of BGP attacks. *Computer Communications Review*, 34(2):1–8, Apr. 2004.

[33] Office of the President of the United States. Priority II: A National Cyberspace Security Threat and Vulnerability Reduction Program. National Strategy to Secure Cyberspace, Nov. 2004.

[34] R. Perlman. *Network Layer Protocols with Byzantine Robustness*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, Oct. 1988. MIT/LCS/TR-429.

[35] J. Postel. Internet Protocol. RFC 791, Sept. 1981.

[36] J. Postel. Transmission Control Protocol - DARPA Internet Protocol Program Specification. Sept. 1981. RFC 793.

[37] J. Puig, M. Achemlal, E. Jones, and D. McPherson. Generic Security Requirements for Routing Protocols, July 2004.

[38] Y. Rekhter and P. Gross. Application of the Border Gateway Protocol in the Internet. RFC 1772, Mar. 1995.

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

[40] J. Rexford, J. Wang, Z. Xiao, and Y. Zhang. BGP routing stability of popular destinations. In *IMW '02: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, pages 197–202, New York, NY, USA, 2002. ACM Press.

[41] R. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, Feb. 1978.

[42] K. Seo, C. Lynn, and S. Kent. Public-Key Infrastructure for the Secure Border Gateway Protocol (S-BGP). In *IEEE DARPA Information Survivability Conference and Exposition II*, June 2001.

[43] B. Smith and J. Garcia-Luna-Aceves. Securing the Border Gateway Routing Protocol. In *Proceedings of IEEE Global Internet 1996*, London, UK, Nov. 1996.

[44] L. Subramanian, V. Roth, I. Stoica, S. Shenker, and R. Katz. Listen and Whisper: Security mechanisms for BGP. In *Proceedings of NSDI'04*, Mar. 2004.

[45] S. Teoh, K. Ma, S. Wu, D. Pei, L. Wang, L. Zhang, D. Massey, and R. Bush. Visual-Based Anomaly Detection for BGP Origin AS Change (OASC) Events. In *Proceedings of IEEE/IFIP DSOM '03*, October 2003.

[46] P. C. van Oorschot, T. Wan, and E. Kranakis. On Interdomain Routing security and Pretty Secure BGP (psBGP). *ACM Transactions on Information and System Security (TISSEC)*, 10(3), 2007.

[47] L. Wang, X. Zhao, D. Pei, R. Bush, D. Massey, A. Mankin, S. F. Wu, and L. Zhang. Protecting BGP Routes to Top Level DNS Servers. In *Proceedings of the 23rd International Conference on Distributed Computing Systems (ICDCS)*, May 2003.

[48] R. White. Deployment considerations for secure origin BGP (soBGP). Internet Draft, Oct. 2002.

[49] K. Zhang, S.-T. Teoh, S.-M. Tseng, C.-N. Chuah, K.-L. Ma, and F. Wu. Performing BGP experiments on a semi-realistic Internet environment. North American Network Operators Group (NANOG), October 2004.

[50] X. Zhang, S. Wu, Z. Fu, and T.-L. Wu. Malicious Packet Dropping: How It Might Impact the TCP Performance and How We Can Detect It. In *Proceedings of ICNP 2000*, Nov. 2000.

[51] Y. Zhang, Z. Zhang, Z. M. Mao, Y. C. Hu, and B. M. Maggs. On the Impact of Route Monitor Selection. In *Proceedings of the ACM Internet Measurement Conference (IMC)*, San Diego, CA, USA, Oct. 2007.

[52] M. Zhao, S. W. Smith, and D. M. Nicol. Aggregated path authentication for efficient BGP security. In *Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS'05)*, Nov. 2005. Alexandria, VA, USA.