Aggregated Path Authentication for Efficient BGP Security

Meiyuan Zhao[*]
Dept. of Computer Science
Dartmouth College
Hanover, NH 03755
zhaom@cs.dartmouth.edu

Sean W. Smith
Dept. of Computer Science
Dartmouth College
Hanover, NH 03755
sws@cs.dartmouth.edu

David M. Nicol
Dept.of Electrical and
Computer Engineering
University of Illinois
Urbana, IL 61801
nicol@crhc.uiuc.edu

ABSTRACT

The *Border Gateway Protocol (BGP)* controls inter-domain routing in the Internet. BGP is vulnerable to many attacks, since routers rely on hearsay information from neighbors. *Secure BGP (S-BGP)* uses DSA to provide route authentication and mitigate many of these risks. However, many performance and deployment issues prevent S-BGP's real-world deployment. Previous work has explored improving S-BGP processing latencies, but space problems, such as increased message size and memory cost, remain the major obstacles. In this paper, we design *aggregated path authentication* schemes by combining two efficient cryptographic techniques—signature amortization and aggregate signatures. We propose six constructions for aggregated path authentication that substantially improve efficiency of S-BGP's path authentication on both speed and space criteria. Our performance evaluation shows that the new schemes achieve such an efficiency that they may overcome the space obstacles and provide a real-world practical solution for BGP security.

Categories and Subject Descriptors

C.2.0 [**Computer-communication networks**]: General-security and protection

General Terms

Design, Security, Performance

Keywords

routing, security, BGP, authentication, performance

1. INTRODUCTION

The *Border Gateway Protocol (BGP)* [38] is a distributed routing protocol that establishes how Internet traffic is routed between *autonomous systems (ASes)*. The stability of the Internet relies on the correct and efficient functioning of the BGP protocol. Despite its central role in the Internet, BGP lacks security [41]. The root of the problem is that BGP relies on hearsay information to update routing tables. Malicious routers can insert false information into the messages they send, which will be used by other routers and further be propagated when honest routers send extensions of these forged messages. A successful compromise of a router may cause many serious security problems quickly throughout the Internet [34].

The current primary security proposal, *Secure BGP (S-BGP)* [24], addresses two major BGP vulnerabilities—lack of authenticity of the information conveyed in messages and lack of authorization for BGP routers to represent certain ASes. S-BGP ensures that a propagated route is authorized by each AS in the path.

When it comes to deployment, S-BGP has faced two main obstacles: time and space. Performing the signing and verification the protocol requires can significantly slow down the time it takes for route changes to propagate throughout the network. To make S-BGP perform fast, routers require large amounts of space to store the necessary public key certificates and digital signatures [25]—indeed, space for the digital signatures for stored route announcements has been cited as being the main obstacle to S-BGP deployment [21]. Previously, we proposed a *Signature Amortization (S-A)* scheme [35] that can significantly speed up S-BGP route authentication, but at the cost of even higher memory consumption. In subsequent work, we considered using aggregate signatures, which reduced memory but greatly increased authentication time—even more than original S-BGP [46].

In this paper, we propose new *aggregated path authentication* schemes for authenticating path information in BGP route announcements. The main idea is to combine the time-efficient scheme of signature amortization with the space-efficient techniques of aggregate signatures [4]. The aggregate signature techniques effectively reduce the number of stored signatures necessary for path authentication. Furthermore, we show that they are capable of producing very short signatures and effectively utilizing hardware acceleration to significantly speed up signature verification.

We propose six constructions of aggregated path authentication to take advantage of options provided by S-A and aggregate signature techniques. We then evaluate their performance using network simulation. Mainly, compared with aggressively optimized S-BGP, the aggregated path authentication schemes achieve significant performance enhancements:

- Software-only implementations can reduce convergence time by 32% over S-BGP.

[*]As of October 2005, her affiliation will be Intel Corporation.

- With hardware acceleration, we can reduce processing latency by 68%. The resulting convergence time is very close to the speed of original BGP without any cryptographic overhead.
- We can shorten the message size by 66% on average. Furthermore, the amount of signature data per message stays roughly constant with the path length.
- We can reduce the signature memory requirement by more than 72%, even when routers try to cache all information. This suggests an overall memory savings of 60–67%.

Given the huge improvements on space and improvement on timing (particularly if we use hardware to essentially eliminate the timing cost), aggregated path authentication schemes remove major obstacles to deploying S-BGP. We are one step closer to practical and efficient BGP security.

In this paper, Section 2 reviews BGP and S-BGP path authentication. Section 3 reviews related work on improving BGP security performance. Section 4 presents our new aggregated path authentication schemes. Section 5 presents performance evaluation methodology and experiment results. Section 6 further discusses real-world deployment issues, and Section 7 concludes this study.

2. BGP AND S-BGP

The Border Gateway Protocol (BGP) is the routing protocol for maintaining connectivity between autonomous systems (ASes) in the Internet. Each AS is assigned a unique integer as its identifier, known as its *AS number*. An AS manages subnetworks expressed as IP *prefixes* (ranges of IP addresses). A BGP *speaker* (a router executing BGP protocol) maintains connections called *BGP sessions* with its *peers* (neighboring speakers), and sends an Update to announce a new preferred route to prefix p. The route is composed of attributes. The two most important attributes are the prefix and the AS path. The *AS path* is a sequence of AS numbers that specifies a sequence of autonomous systems through which one can traverse the network; the last AS in this sequence is the *originator* of this route. For instance, if the autonomous system AS_k owns IP prefix p, the autonomous system AS_0 might send out an Update $(p, \{AS_0, AS_1, \ldots AS_k\})$ to announce the route that AS_0 prefers to use for reaching p. BGP speakers keep routes in the *Routing Information Bases (RIBs)*: one Adj-RIBs-In per peer keeps received routes from the peer; one Adj-RIBs-Out per peer stores all sent routes to that peer; and Loc-RIB records all preferred routes for each prefix. Depending on implementation, the Adj-RIBs and the Loc-RIB can either be separated or centralized physically. In this study, we consider the Adj-RIBs-In and Loc-RIB together as the *routing table* for the BGP speaker.

Typically, a speaker's Loc-RIB changes when it adds a new route, deletes a preferred route, or replaces a previously preferred route with a new one. BGP speakers incrementally send Update messages to their peers to announce such changes. When speakers establish (or re-establish) a BGP session, they share their own Loc-RIBs with each other via a large number of Update messages announcing all preferred routes. Processing of an Update message may create a number of new Updates, since an Update may result in new preferred routes. If the speaker chooses to announce a new preferred route, it extends the existing AS path by prepending its AS number to it and sending this extension to all of its peers, except the one who sent the route earlier. When a speaker announces a route to prefix p, it implicitly *withdraws* the last route it announced to p. The recipient, understanding this implicit route withdrawal, decides whether it prefers the new route. A withdrawal can also be an explicit announcement, with no mention of an alternative preferred route. In this case, the recipient may examine the previously received routes from Adj-RIBs-Ins to the same prefix and decide whether there is an alternative to announce to its peers. If no such route found at hand, it simply withdraws the route as well.

BGP rate-limits the sending of Update messages with parameter called the *Minimum Route Advertisement Interval (MRAI)*, which is basically the minimum amount of time that must elapse between successive batches of Updates sent to a peer. BGP speakers have output buffers to keep waiting Update messages, and send them in batches when reaching the MRAI. A speaker may have a different MRAI for each of its peers or may have one MRAI that controls all peers. In practice, throughout the Internet, the default value of the MRAI is 30 seconds.

Any change of network reachability will be reflected in the Loc-RIB of some BGP speaker. BGP will then propagate this change via Update messages through the entire network, like a wave. The *convergence time* measures the length of time for this wave of announcements to die out completely—in other words, for the network to reach a stable state. During the transient period before convergence, the continual changing of preferred routes degrades the effectiveness of packet forwarding. Longer convergence times thus reflect increased network instability and may cause severe network performance problems. Studies of BGP [18, 27, 29] have considered convergence and possible optimizations to control and accelerate it.

Security. It has been widely recognized that the lack of security in BGP is a critical problem to the Internet [24, 34]. BGP is vulnerable to malicious actors as well as to accidental errors. Because BGP speakers completely rely on and believe in the routing information sent from peers, authentication mechanisms are needed to provide route announcement authenticity. *Origin authentication* considers whether the originating AS really controls the claimed IP address ranges. *Path authentication* confirms that all the ASes are authorized to announce the routes to the destination IP address block(s). The entire AS path is authenticated only when the all participating ASes are confirmed to propagate the AS paths honestly and to attach correct extensions to the AS paths correctly.

The primary security proposal, *Secure BGP (S-BGP)* [24] uses attestations to authenticate route announcements. *Address Attestations (AAs)* are for origin authentication. *Route Attestations (RAs)* are for path authentication. To support signing and verification operations, S-BGP also sets up public key infrastructures to establish the authenticity of the involved parties. ASes, organizations, and speakers have their own X.509 public key certificates [20], expressing the binding between the public key and the identity. There are also address allocation certificate expressing authorized IP address ownership by organizations.

Next, we review more details of the S-BGP design. Since the focus of this paper is on path authentication, we skip discussions on S-BGP PKIs and address attestations; see [24] for more details. For performance comparison, we assume all path authentication schemes discussed in this paper use the same PKI systems and AAs by S-BGP. Their difference is only on their designs for path authentication.

For path authentication, a *route attestation (RA)* is signed by a BGP speaker to authenticate the existence and position of an AS number in an AS path [24]. Such attestation is nested: each BGP speaker signs the AS path in sequence, as it joins. That is, first the origin BGP speaker signs the AS number of the origin autonomous system, the prefix, and the intended receiver (in the form of AS number). The next signer is the recipient of this RA; it computes

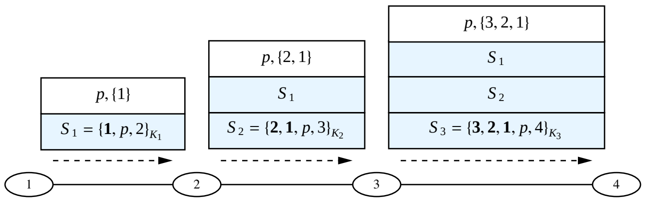
$p, \{1\}$
$S_1 = \{\mathbf{1}, p, 2\}_{K_1}$
$p, \{2, 1\}$
S_1
$S_2 = \{\mathbf{2}, \mathbf{1}, p, 3\}_{K_2}$
$p, \{3, 2, 1\}$
S_1
S_2
$S_3 = \{\mathbf{3}, \mathbf{2}, \mathbf{1}, p, 4\}_{K_3}$
1
2
3
4

Figure 1: This figure—adapted from [46]—sketches the process of sending route announcements and their route attestations. We have four ASes numbered as 1, 2, 3, and 4. AS 1 initiates the process by sending announcement (p, {1}) stating that it owns prefix p and it is reachable. It generates the corresponding route attestation by signing {1, p, 2} using its private key K_1. It puts its AS path first, then the prefix, then the intended recipient. The other ASes continue this process by signing the latest AS paths, the prefix, and the intended recipient. The figure shows the AS path components in bold.

and signs the concatenation of the new AS path, the prefix, and intended recipient. The process goes on until the entire AS path is signed. The inclusion of the intended recipient and the prefix in each signature is necessary to prevent against "cut-and-paste" attacks. Figure 1 demonstrates the process of sending route announcements and their route attestations using an example.

Performance Issues. Several factors affect the performance of path authentication in S-BGP, given the structural properties of RAs.

First, BGP speakers consume extra CPU cycles when signing and verifying RAs and when handling and validating certificates. Each preferred route announcement involves one signing operation by each signer and k verification operations by each verifier (where k is the number of RAs for this AS path). Second, RAs increase message size. Each message with an AS path of length k carries k nested RAs. Finally, to decrease the number of signing/verification operations, one could cache the signed/verified routes in memory. Memory cost becomes another important performance issue. Furthermore, speakers should also know public keys, thus corresponding public key certificates, to validate RAs. To operate fast, speakers hold caches in memory as much as possible.

Researchers have introduced a number of optimizations for S-BGP [23], mainly focusing on caching signed and verified routes and applying DSA pre-computation. These optimizations reduce the computational cost related to cryptographic operations, in exchange for extra memory cost and computation complexity.

The performance problems of S-BGP path authentication hinder its realization on the Internet. The criticisms are mainly focused on too much memory for holding signatures and certificates for Update processing, and high processing overheads. This research solves both problems; as the result, we bring S-BGP one step closer to real-world realization.

3. RELATED WORK

The performance studies in [23, 25] offer detailed discussions on deploying S-BGP in the real world. The authors collected a variety of data sources to analyze S-BGP's performance impacts on BGP processing, transmission bandwidth, and routing table size. These studies concluded that the memory requirements of holding route information, the necessary certificates, and related cryptographic data are the major hurdle. On the other hand, our previous studies in [35, 46] examined S-BGP performance using simulation. Due to public key cryptography, S-BGP path authentication has high operational latency and thus greatly increases BGP convergence time. We then proposed the signature amortization (S-A) scheme for improving efficiency. However, the substantial speed-up by S-A comes at the cost of significant lengthened messages and increased memory consumption.

Other studies tried to improve efficiency by lowering the security requirements. *psBGP* [43] increases practicality by combining the centralized model of authenticating AS numbers with a decentralized model of authenticating IP prefix ownership and AS paths. Subramanian et al. [42] proposed the *Listen* and *Whisper* protocols to address the BGP security problem. The Listen protocol helps data forwarding by detecting "incomplete" TCP connections; the Whisper protocol uncovers invalid route announcements by detecting inconsistency among multiple Update messages originating from a common AS. The Listen and Whisper approach does not rely on any centralized database. It aims for "significantly improved security" rather than "perfect security".

There are efforts on securing BGP using symmetric key algorithms [15, 22]. These proposals are more efficient on speed, but require more storage, loose time synchronization, and/or complex key-pair pre-distribution. Besides cryptographic authentication approaches, some studies used database and system monitoring approaches to secure BGP routing operations [14, 42, 44]. Other work has examined the origin authentication problem [1, 44].

4. DESIGN OF AGGREGATED PATH AUTHENTICATION SCHEMES

In this study, our goal is to maintain the strong security that S-BGP provides, while also providing more efficient path authentication schemes that ease practical deployment on the Internet. We target two problems: improving the processing latency and reducing the space burden on the routers and on the network. We propose *aggregated path authentication (APA)* schemes that combine our time-efficient technique of signature amortization (S-A) [35] with the space-efficient cryptographic technique of aggregate signatures [4, 5, 28].

In the first step, we use signature amortization techniques to reduce the latency by cryptographic operations, thus speeding up BGP convergence. Next, we use aggregate signature algorithms to reduce Update message size and signature memory requirements on BGP speakers, by using an aggregated signature for authenticating the entire AS path. This idea has been mentioned in conversation as a potential application by the designers of aggregate signature schemes [4, 5]. In previous work, we applied sequential aggregate signatures to S-BGP [46], but space consumption only got worse. We now take a more thorough look.

In the rest of this section, we first briefly introduce the signature amortization technique and the aggregate signature algorithms. Then, we discuss the details of our new efficient aggregated path authentication schemes.

Signature Amortization (S-A). Our previous study [35] proposed Signature amortization (S-A), specifically for speeding up the processing of S-BGP RAs. S-BGP uses DSA as its signature algorithm, mainly because DSA produces short signatures. However, DSA signature verification is relatively slow. Realizing that majority of cryptographic operations involved are signature verifications, we used RSA, which has fast verification but slow signings. We then compensated for the expensive RSA signings by amortization, in two steps.

In step one, when a BGP speaker sends the same route announcement to multiple recipients, S-A collapses it to literally the same announcement: using a bit vector (or a more space-efficient equivalent) to express which of the speaker's peers are the recipients. Thus, the speaker only needs to generate one signature, instead of one for each recipient; the verifier of this RA uses the bit vector to check the intended receiver. To do this, the speaker needs to pre-establish an ordered list of its neighbors and distribute this list to potential verifiers; S-A achieves the goal by putting this information in the speaker's X.509 certificate.

BGP speakers keep outgoing Update messages in output buffers and, using MRAI timers, send them in bulk. Thus, in step two, when an MRAI timer fires and a BGP speaker sends the messages accumulated in its output buffers, it collects all "unsigned" messages, builds a Merkle hash tree [30] on them, and signs the root of the tree—thus generating one signature for all unsigned messages, instead of one for each message. A leaf of the tree is the hash of the pair of a route and the recipient bit vector. The resulting RA consists of the RSA signature on the root, the hash path from the root to that leaf, the route, and the recipient bit vector. A verifier of the RA can use these hash values and information in the route announcement to reconstruct the root of the tree correctly. There are trade-offs, however. The verifier needs to perform a few extra hashing operations when verifying a RA, and the message size grows (due to the hash path).

Our studies [35, 46] have shown that S-A is very efficient in terms of speed. However, it substantially increases message size and memory cost. This is because RSA signatures are much longer than DSA signatures (128 bytes vs. 40 bytes, for the RSA modulus length currently regarded as secure), and S-A needs additional hash values for each signature verification (20 bytes each, assuming SHA-1 is still secure). It is suggested that an S-A variant can use only the bit vectors to amortize the signing cost. This variant can decrease space cost, since no hash paths are involved. Moreover, because bit vectors are much more stable than the Merkle hash trees, it provides the opportunity to cache the signed and verified routes. Experiments in [46] have shown that this variant reduces the message size, but increases the convergence time compared with original S-A design.

Aggregate Signatures. An *aggregate signature* [4] is a digital signature that supports aggregation: given n signatures on n distinct messages by n distinct users using an aggregate signature algorithm, it is possible to aggregate these signatures into a single short signature. This single signature (and the n original messages) will convince the verifier that the n users did indeed sign the n original messages. Next, we introduce details of two main proposals of constructing aggregate signatures.

The *general aggregate signature* approach is based on a co-GDH signature scheme, which can be based on any gap group. The short signature scheme by Boneh, Lynn, and Shacham *(BLS)* [7] is one such co-GDH signature scheme that makes use of elliptic curves. It is referred to as a "general" aggregate signature, since the aggregation algorithm is public—given n signatures $\sigma_1, \ldots, \sigma_n$, anyone can use a public aggregation algorithm to calculate the aggregate signature σ. The aggregation algorithm is implemented by using a bilinear map between two (multiplicative) cyclic groups of prime order p, G_1 and G_2. Given an additional group G_T, such that $|G_1| = |G_2| = |G_T|$, a bilinear map is a map $e : G_1 \times G_2 \to G_T$ with the following properties:

- Bilinear: for all $u \in G_1, v \in G_2$ and $a, b \in \mathbb{Z}, e(u^a, v^b) = e(u, v)^{ab}$.
- Non-degenerate: $e(g_1, g_2) \neq 1$, where g_1 is a generator of G_1 and g_2 is a generator of G_2.

We briefly summarize *Sign*, *Verify*, *Aggregate*, and *Aggregate Verify* algorithms as the following. Let H be a hash function.

- **Sign** For a particular user, the algorithm works a normal co-GDH signature scheme. Given the private key x and a message M, compute $h \leftarrow H(M)$, where $h \in G_2$, and $\sigma \leftarrow h^x$. $\sigma \in G_2$ is the signature.
- **Verify** Given a user's public key v, the message M, and the signature σ, compute $h \leftarrow H(M)$; accept if $e(g_1, \sigma) = e(v, h)$ holds.
- **Aggregate** For a set of users U, where $|U| = k$, signatures $\{\sigma_i \in G_2 \mid 1 \leq i \leq k\}$ on messages $\{M_i \mid 1 \leq i \leq k\}$, compute $\sigma \leftarrow \prod_{i=1}^{k} \sigma_i$. The aggregate signature is $\sigma \in G_2$.
- **Aggregate Verify** Given the aggregate signature σ, the message set $\{M_i \mid 1 \leq i \leq k\}$ on which it's based, and public keys $v_i \in G_1$ for all users $u_i \in U$, verify the aggregate signature σ in two steps:
  1. ensure that the messages M_i are all distinct, otherwise reject; and
  2. compute $h_i \leftarrow H(M_i)$ for $1 \leq i \leq k$, and accept if $e(g_1, \sigma) = \prod_{i=1}^{k} e(v_i, h_i)$ holds.

Like a co-GDH signature, a bilinear aggregate signature is a single element of G_2. Note that the aggregation can be performed incrementally. This way, the aggregation is as fast as a modular multiplication. The calculations for verify and aggregate verify algorithms are mostly the mapping e, which can be implemented using pairing calculations. We discuss more details on how to compute pairing efficiently later in Section 5.1.3.

In contrast, a *sequential aggregate signature* scheme [28] is based on homomorphic trapdoor permutations, such as RSA. Each signer incrementally signs the new message and incorporates it into the aggregate signature σ. A party with knowledge of n messages, public keys of the n ordered signers, and σ is able to verify that each signer s_i has correctly signed his message M_i and σ is a valid sequential aggregate signature. The designers also showed how to instantiate the construction with the RSA trapdoor permutation. Briefly, we review the resulting RSA aggregate signature scheme for n users with moduli of length l in the following. Let $H : \{0,1\}^* \to \{0,1\}^{l-1}$ be a hash function. Note that the following version requires that the moduli must be ordered. However, the moduli can be unrestricted with a few additional steps in the algorithm.

- **Key generation** Each user i generates an RSA public key (N_i, e_i) and private key (N_i, d_i).
- **Aggregate Sign** `AggrSign`(σ', M_i). As the base case, let $i = 1$, and the initial aggregate signature $\sigma' \leftarrow 0$ on an empty message set. For the ith signer, given a valid aggregate signature σ' on previous $i-1$ messages $\{M_1, \ldots, M_{i-1}\}$ and public keys, she computes
  $h_i \leftarrow H((M_1, \ldots, M_i), ((N_1, e_1), \ldots, (N_i, e_i)))$, $y \leftarrow h_i + \sigma'$, and outputs $\sigma \leftarrow y^{d_i} \mod N_i$.
- **Aggregate Verify** Given aggregate signature σ on i messages $\{M_1, \ldots, M_i\}$, and public keys, the verifier does the following checks:
  1. public keys satisfy requirements;
  2. check that $0 \leq \sigma \leq N_i$;
  3. if $\gcd(\sigma, N_i) = 1$, let $y \leftarrow \sigma^{e_i} \mod N_i$, else let $y \leftarrow \sigma$;
  4. compute $h_i \leftarrow H((M_1, \ldots, M_i), ((N_1, e_1), \ldots, (N_i, e_i)))$ and $\sigma' \leftarrow (y - h_i) \mod N_i$;
  5. verify σ' recursively;
  6. accept if $\sigma = 0$ holds when $i = 0$, reject otherwise.

The details of aggregate sign algorithm is the same as the ordinary RSA signing algorithm plus a few additional big number calculations, whose running times are negligible compared with ordinary RSA computations. Hence, we estimate the running time of aggregate sign as the same as the ordinary RSA signing. The recursive aggregate verify algorithm works through layers of the aggregation until the base case. Thus the running time can be estimated as the time of i RSA verifications plus a bit time on extra calculation.

Aggregated Path Authentication. The aggregated path authentication schemes combine S-A with aggregate signature schemes. Again, S-A is a good candidate to speed up processing Updates, while aggregate signatures can shorten the message size and release the memory burden. The number of signatures in a route announcement is no longer linear in the length of the AS path. One aggregate signature is enough to the authenticity of the entire AS path.

Using S-A, we have two choices. Recall that S-A amortizes signing cost in two steps. We may choose to use bit vectors only (referred to as *S-A-vector*) or to apply bit vectors with Merkle hash trees together (referred to as *S-A-tree*). Although S-A-vector may not result in high degree of signing amortization, it has two advantages. First, since speakers no longer enclose hash paths in Update messages to help signature verification, we can certainly further reduce the space cost. Second, S-A-vector provides much more stable signatures for routes. In other words, with high probability, when the same route announcement has been signed twice and sent to the same set of recipients, it will have the same signature. This property allows the speakers to cache verified signatures to avoid duplicated cryptographic operations. In contrast, such straightforward caching optimization for the S-A-tree is useless. If the speaker uses the S-A-tree scheme, it is unlikely that the same route announcement will end up with the same signature since the tree construction depends on the outgoing routes in the output buffers. Such information is highly dynamic.

As discussed earlier, we also have two choices on aggregate signature schemes. General aggregate signatures are based on the BLS short signature scheme. For standard security parameters, the signature length is about half that of a DSA signature with a similar level of security. Sequential aggregate signatures are implemented using RSA trapdoor permutation, thus the signature length is the same that of a RSA signature with the same level of security. Certainly, general aggregate signatures outperform in terms of space. However, the aggregate verify operation provided by the sequential aggregate signature scheme may be substantially faster than the one provided by general aggregate signature scheme. Since the majority of cryptographic operations performed by BGP speakers are verifications, sequential aggregate signature scheme can potentially be the winner in terms of the speed.

To achieve efficient security, we design four constructions by combining choices we have for S-A and aggregate signatures. We then use network simulation to evaluate their performance and to identify the most efficient scheme.

We define several notations. Let (pa, p) be the current route announcement, where p is the announced prefix and pa is the associated AS path. Let v be the bit vector indicating the recipients. Let σ' be the aggregate signature on the previous route announcement and σ be th newly generated aggregate signature on the updated route announcement. Let $a \parallel b$ stand for concatenating a with b. We now consider the APA constructions.

GAS-V BGP speakers organize outgoing route announcements using a bit vector v, generate BLS signature $s \leftarrow sign(pa \parallel p \parallel v)$, and compute the aggregate signature $\sigma \leftarrow \sigma' \cdot s$. The outgoing route announcement contains the route, the bit vector, and the aggregate signature, $\{(pa, p, v), \sigma\}$. For caching, speakers can cache the route (pa, p, v) and associated aggregate signature σ to avoid duplicated signing or verification. Furthermore, we can use either software or hardware implementation of pairing calculation for verify and aggregate verify operation. We denote these two variants as *GAS-V(SW)* and *GAS-V(HW)*.

GAS-T BGP speakers organize outgoing route announcements using a bit vectors, construct a hash tree (the resulting root of the tree is R), generate BLS signature $s \leftarrow sign(R)$, and compute the aggregate signature $\sigma \leftarrow \sigma' \cdot s$. The outgoing route announcement contains the route (pa, p), the bit vector v, the hash path in the tree, and the aggregate signature σ. BGP speakers do not cache any signed or verified routes, their aggregate signatures, or hash paths. Similar to GAS-V, there are two variants for GAS-T—*GAS-T(SW)* and *GAS-T(HW)*.

SAS-V BGP speakers organize outgoing route announcements using a bit vector v, and generate aggregate signature σ using the function $\texttt{AggrSign}(\sigma', pa \parallel p \parallel v)$. The outgoing route announcement contains the route, the bit vector, and the aggregate signature, $\{(pa, p, v), \sigma\}$. For caching, speakers can cache the route (pa, p, v) and associated aggregate signature σ to avoid duplicated signing or verification.

SAS-T BGP speakers organize outgoing the route announcements using bit vectors, construct a hash tree (the root of the tree is R), and generate aggregate signature $\sigma \leftarrow \texttt{AggrSign}(\sigma', R)$. The outgoing route announcement contains the route (pa, p), the bit vector v, the hash path in the tree, and the aggregate signature σ. BGP speakers do not cache any signed or verified routes, their aggregate signatures, or hash paths.

5. PERFORMANCE EVALUATION

We evaluate APA schemes using network simulation. The goal of simulation is to compare aggregated path authentication schemes with S-BGP route attestations. We focus on simulation methodology and experiment results in this section. Section 6 will extend our discussion on real-world deployment issues.

5.1 Evaluation Methodology

The complexity of the Internet makes an analytical approach difficult; and the reality of Internet makes empirical approaches also unworkable. Thus we use simulation. Section 5.1.1 describes the metrics we use for performance comparison. Section 5.1.2 discusses the tools we use to carry out experiments. Section 5.1.3 presents issues of getting appropriate benchmarks of running times for various cryptographic primitives.

5.1.1 Performance Metrics

We use a set of metrics to evaluate performance in terms of time and space.

For time, we measure the number of cryptographic operations involved, the resulting CPU cycles, and the BGP convergence time: the time it takes the system to re-achieve a stable state after a perturbation, such as a new route announcement, a route withdrawal, or a router reboot. Particularly in our experiments, we measure rebooting convergence time—the time between when a crashed BGP speaker returns to life and all the changes that induces through the network. For each security scheme, we compare its convergence time with convergence time that original BGP achieves for

Algorithms	Running Time (*ms*)
Miller's Algorithm on $\mathbb{F}_{3^{97}}$	24.0
BKLS on $\mathbb{F}_{3^{97}}$	23.6
Refined Duursam-Lee on $\mathbb{F}_{3^{97}}$ [17]	16.8
Modified Duursam-Lee on $\mathbb{F}_{3^{97}}$ [3]	8.6
Hardware implementation [26]	1.3

Table 1: Running times of Tate pairing calculation. Running times by software implementations are normalized to 1 GHz processor. The hardware implementation assumes a conservative 10 MHz clock frequency on the target technology.

the same perturbation. Given the distributed nature of BGP, convergence time is very difficult to predict using analytical techniques.

For space, we measure both the message size and the storage cost in memory. For simulation, we assume a simple form of Update messages. We measure the bytes for basic Update message fields and bytes for additional signatures, bit vectors, and hash values. Note that current MTU limitation on Update message is 4096 bytes. In experiments, we relax this limitation. The experiments report both average and maximum message sizes for us to understand the efficacy of different options.

To understand memory cost, we focus on signature memory requirements. That is, experiments report memory space for route announcements, signatures and bit vectors. For fair comparison, we assume BGP speakers spend same amount of memory for storing certificates and AAs. Our further discussion in Section 6 covers more issues related to storing certificates and AAs.

5.1.2 *Simulation*

Similar to our other studies [35, 46], our experiments use a network simulator SSFNet [36], a discrete-event simulator that provides a comprehensive model of basic BGP operations [37]. We take advantage of the added hooks for variants of processing models of BGP security schemes [35].

Throughout this study, we evaluate security schemes in the same network topology and same BGP activity setting. We use a 110-AS topology, with one operating BGP speaker per AS. For modeling simplicity, each BGP speaker announces two prefixes. In our model, each AS also possesses *virtual BGP speakers* that don't actually run a simulated BGP protocol. We use the number of such BGP speakers to represent the size of an AS; its size affects the time it takes for one Update message to be propagated through an AS.

We use the public data provided by the RouteViews project [39] to generate a graph of AS connectivity of the Internet, and then reduce the size to 110 ASes using a collapsing procedure. This reduced graph still preserves certain macroscopic properties seen on the Internet [10]. We further incorporate our estimation of route filtering policies into the topology using a similar method proposed in [13].

During normal BGP activities, we let one BGP speaker crash and come back to life. We evaluate the performance of the entire system during router rebooting process. The workload on BGP speakers could be much higher than normal BGP activities. When re-establishing BGP sessions with its peers, the rebooting BGP speaker receives routing table dumps in a short period of time from each its peers, via a large amount of route announcements. To maximize the effects, we let the rebooting BGP speaker to be the one with the most peers.

5.1.3 *Benchmarks*

It's straightforward to decide the unit length of data structures involved in path authentication. For a similar level of security, S-BGP uses DSA algorithm with SHA-1, which results in 40-byte signatures and 20-byte hash values. A BLS short signature is 20 bytes long, and thus so is the aggregate signature by the general aggregate signature scheme. An RSA signature is 128 bytes long, which is also the length of a sequential aggregate signature. In building the hash trees, S-A uses SHA-1, per standard usage. (Exploring shorter hashes is an area for future work.)

We obtained the running times for standard signature algorithms, such as RSA and DSA, by benchmarking the OpenSSL crypto library. However, OpenSSL does not have implementations of aggregate signatures. Fortunately, we can decompose the calculation of each algorithm, obtain the running time of each step, and combine to estimate the total running time. We consulted the community and the literature for other crypto libraries and implementations [2, 6, 32].

As we have discussed, the implementation of sequential aggregate signatures just requires minor modifications to the RSA algorithm. Hence, we immediately estimate that signing and verification time by SAS is the same as the RSA algorithm (since the deltas introduced by additional arithmetic operations are negligible); and aggregate verification on k distinct messages costs about k times individual verification times.

To understand the running times by general aggregate signatures based on elliptic curves, we turned to the literature of pairing-based cryptosystems. From [2], we learn that it costs about 2.2 *ms* on a 1GHz processor to sign a message using BLS. Aggregation on two BLS signatures needs another modular multiplication on 157-bit numbers, whose running time is negligible compared with signing.

To estimate verification and aggregation verification performance, we need to understand pairing calculation. Pairing calculation in the verification and aggregate verification operations is relatively slow compared with scalar point multiplication in signing operations. In the general aggregate signature scheme, one verification is composed of two pairing calculations, and an aggregate verification on k distinct messages requires $k+1$ pairing calculations. In recent years, an ever-increasing number of pairing-based cryptosystems have appeared in the literature, driving research into efficient algorithms for the implementation of bilinear pairing on elliptic curves. To date, the Tate pairing [11] has attracted attention as the most efficiently computable bilinear pairing on elliptic curves. In particular, Tate pairing over supersingular elliptic curves achieves its maximum security in characteristic three. The classic algorithm for Tate pairing computation on elliptic curves is Miller's algorithm [31]. Later, BKLS/GHS algorithms furthered this development so that the Tate pairing became easier to compute in practice [2, 12]. Duursma and Lee [9] further improved the Tate pairing calculation and extended to more general hyperelliptic curves. Yet even more enhancements to Duursam-Lee Algorithm have appeared for supersingular elliptic curves over fields of characteristic three [3, 40].

Accompanied with dramatic improvements on software implementations of Tate pairing, there are also a few efforts on developing hardware to calculate pairings efficiently [16, 26] The main observation lies in the fact that arithmetic architectures in the extension field $GF(3^{6l})$ are good candidates for parallelization, leading to a similar calculation time in hardware as for operations over the base field $GF(3^m)$ [26]. Table 1 summarizes the running times of pairing calculations. We chose the running times by most efficient software optimization and by hardware acceleration for our simulation experiments. Table 2 illustrates our estimation of running time and signature length. We use these numbers as parameters in the simulation experiments. We assume that routers have 200MHz processors for signature operations in software. The hardware implementation assumes a 10MHz clock frequency.

	1024-bit RSA	1024-bit DSA	1024-bit SAS	GAS based on $\mathbb{F}_{3^{97}}$
Sign (ms)	50.0	25.5	50.0	11.0
Verification (ms)	2.5	31.0	2.5	43.0×2
SW Aggregate Verification (ms)	–	–	$2.5 \times k$	$43.0 \times (k+1)$
HW Aggregate Verification (ms)	–	–	–	$1.3 \times (k+1)$
Aggregate Sign (ms)	–	–	50.0	11.0
Signature length (bytes)	128	40	128	20

Table 2: Benchmarks of signature algorithms with same level of security. Running times are normalized to 200 MHz CPU, a typical type of processors by edge BGP routers on the Internet, except hardware implementation of aggregate verification. We assume that aggregate verification handles k distinct messages. Signature length by general aggregate signature is based on BLS on $\mathbb{F}_{3^{97}}$. For the same level of security, BLS renders 157-bit signatures.

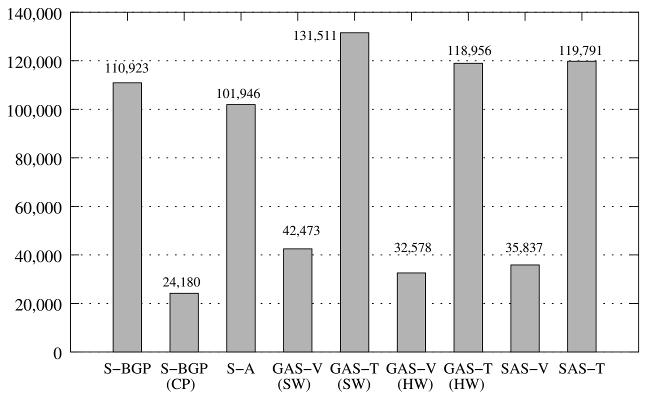

Figure 2: Counts for verify operations. For one aggregated verification on k messages, we count it as k verifications. We also treat the pairing calculation the same way as for normal signature verifications.

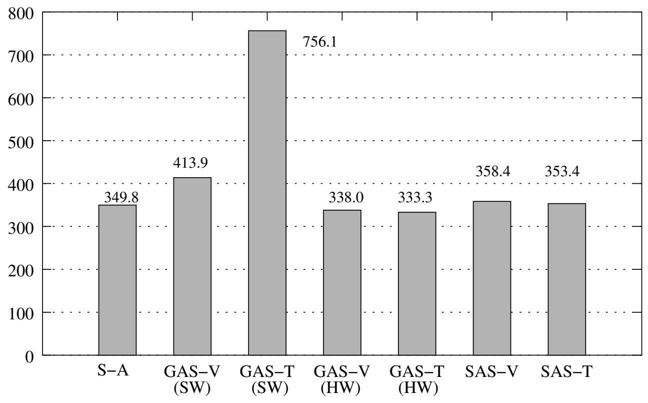

Figure 3: Counts for signing operations.

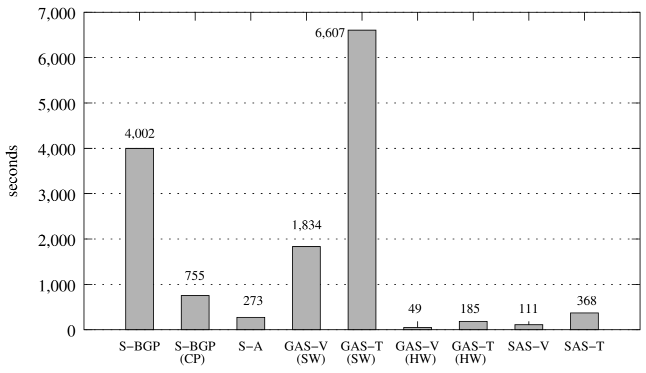

Figure 4: Computational overheads on routers' CPU.

5.2 Simulation Results

The security schemes have performance overhead over the original (unsecured) BGP protocol. Here we are interested in comparing our aggregated path authentication schemes with S-BGP and S-A. Our simulation experiments show that each construction of aggregated path authentication has its strength and weakness. GAS-V(HW) clearly stands out to be the most efficient path authentication scheme. Like S-A, GAS-V(HW) achieves fast BGP convergence, even slightly faster than S-A. At the same time, the resulting messages are about 66% shorter than messages for S-BGP. This keeps the Update message way below the MTU limit. In addition, Update messages essentially do not grow as AS path length increases. This nice property is also reflected in the signature memory consumption. GAS-V(HW) signatures reduce about 72% of memory cost for S-BGP. All together, we can confirm that GAS-V(HW) is a practical and efficient path authentication scheme for BGP that keeps the same level of security as S-BGP route attestations.

As of software-only implementations, GAS-V(SW) has similar memory costs as GAS-V(HW), and still has faster convergence than S-BGP.

5.2.1 Speed

We analyze speed by counting cryptographic operations first, then examining the necessary CPU cycles, and finally comparing convergence time during router rebooting.

In simulation, we model two versions of S-BGP. We use "S-BGP" to denote basic S-BGP, and "S-BGP(CP)" for S-BGP with caching optimization and using DSA pre-computation to speed up the signing process.

Figure 2 and 3 present counting of verification and signing operations. All the schemes in this paper apply the same "lazy verify" optimization proposed by S-BGP [23], that is, the BGP speaker verifies signatures on the route only when it decides to install the route into its Loc-RIB. For verifications, caching optimization is quite effective to reduce the number. As for reducing signing operations, all of the aggregated path authentication schemes are effective. The number of signing operations for S-BGP and S-BGP(CP) are 22,072.3 and 11,521.9 respectively, which are too large to be shown in Figure 3. Most aggregated path authentication schemes as well as S-A can reduce 98% of signings for S-BGP. Surprisingly, GAS-T(SW) is the least efficient construction for amortizing signing cost. The main reason is that GAS-T(SW) verification is

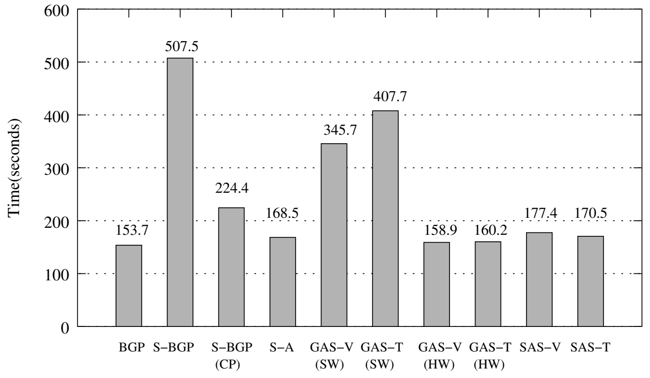

Figure 5: Comparison of convergence time.

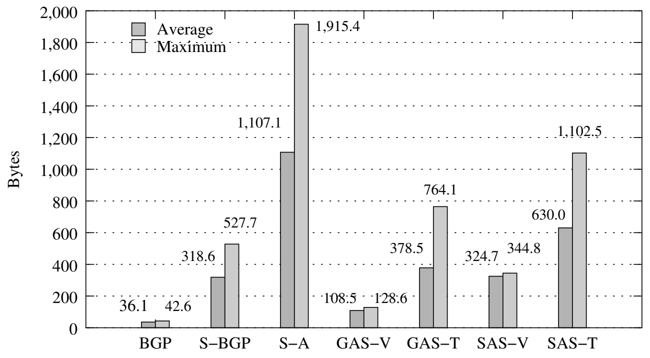

Figure 6: Comparison of message size.

much slower than that of hardware implementation and RSA verification. BGP speakers using GAS-T(SW) spend much longer time processing received routes. As the result, there are not many outgoing messages waiting for MRAI timers at a time. S-A amortizes more than 60 signings to one, while GAS-T(SW) only achieves the amortization degree of 28.2.

Figure 4 presents the total CPU time spent for cryptographic operations. The CPU time for the different aggregated path authentication schemes varies greatly. GAS-V(HW) is the most efficient scheme of our new schemes, while GAS-T(SW) is the worst. The latency is mostly affected by number of operations and unit running times.

As Figure 5 clearly shows, all aggregated path authentication schemes converge faster than S-BGP. All schemes, except general aggregate signatures using software pairing calculation, converge much faster than S-BGP(CP). GAS(HW) convergence even faster than S-A. It is only 5 seconds slower than original BGP without any path authentication mechanism.

Recall that we measure the convergence time during router rebooting process. We can conclude that aggregated path authentication schemes can achieve minimum impact on BGP convergence even when routers are under pressure.

5.2.2 Space

Our experiment results presented in Figure 6 and 7 further conclude that GAS-V(HW) is not only the fastest on convergence time, but is also the most economical on space. Note that GAS-V (or GAS-T) with either software or hardware pairing calculation present similar performance on space. Hence, we simply illustrate experiment results for hardware pairing calculation here. Among all aggregated path authentication constructions, GAS-V produces shortest Update messages. Using GAS-V, we have successfully shorten S-BGP messages by 66%. GAS-V can also save about 72% of the memory requirements by S-BGP.

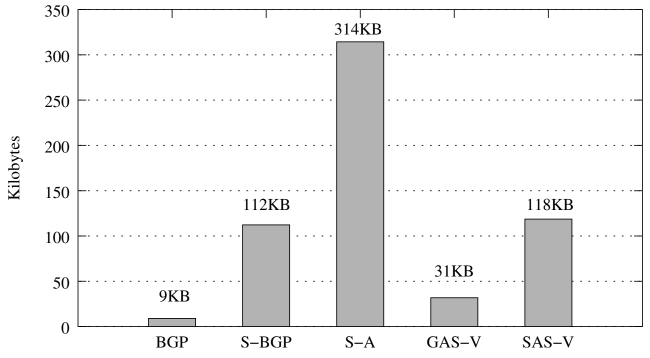

Figure 7: Comparison of memory costs.

The aggregated path authentication schemes are all capable of fast convergence. For the resulting message size, however, tree-based schemes generate longer Update messages, because of extra hash values carried in the messages. Moreover, GAS-based schemes outperform SAS-based schemes on message size, because of much shorter aggregate signatures (20 bytes vs. 128 bytes). Vector-based schemes (GAS-V and SAS-V) have another nice feature: maximum message size is close to the average size. This feature gives us confidence that vector-based schemes will have no difficulty complying with the Update message MTU limit, in the simulated network or real network.

We also compare the memory costs by caching schemes with the original BGP. The numbers shown in Figure 7 are the average memory overhead on one BGP speaker. This include signatures for AAs as well as for RAs. For completeness of experiments, we include the memory overheads on basic routes and AAs for all of the path authentication schemes. GAS-V, again, achieves the best performance. The resulting overhead is only about 28% the amount spent by S-BGP. Note that the current criticism on S-BGP practicality is mainly on extra memory burdens. Our experiments indicate that GAS-V is not just efficient—it can be a practical security solution for BGP path authentication.

6. DISCUSSION

So far, our simulation experiments have confirmed that our design of aggregated path authentication schemes is capable of saving storage space significantly for S-BGP route attestations, and they have minimum impact on BGP convergence even when the BGP speakers are under stress. These results suggest that our aggregated path authentication can be a good candidate for practical deployment in the real world. In doing so, several issues deserve more careful discussion and consideration.

6.1 The Real World

Limited by the scale of the simulation, we are not able to model the entire Internet to study the performance impacts. Using the publicly available BGP data, we could get a sense on what it looks like if an aggregated path authentication scheme is deployed in the Internet and how much it can improve based on S-BGP.

To make our discussion complete, we use the BGP routing table dump from RouteViews to understand the memory cost on a real router. This table dump is collected from AS6447 archive on May 04, 2005, which takes 209MB in MRT [33] format. It contains 162,237 unique IP prefixes and 2,011,005 unique (AS path, prefix) tuples. To cache all received S-BGP route attestations, the router should record 8,284,042 DSA signatures, which requires about 316MB in cache. Moreover, if the router also caches signed address attestations, it should consume another 76.8MB at least. In total, we are looking at 393MB memory cost, adding more than 180% to the BGP routing table size. In fact, we underestimate the amount by examining signature length only. As discussed in [25], the average size would be larger since other information is involved in each attestation. We specifically chose to examine only the signature length, so that we could have appropriate comparison with aggregated path authentication schemes.

Now, let us calculate the memory cost for GAS-V, the most efficient scheme shown in the simulation. We assume that the bit vectors take 4 bytes on average. Since there are 2,011,005 unique (AS path, prefix) tuples, GAS-V will take 108MB memory to cache all received and sent aggregate signatures and bit vectors in memory. This number is substantially smaller than the one for S-BGP. The actual memory cost for signatures only is reduced to less than 29%, increasing the routing table size by only 52%. We believe GAS-V is much more manageable and feasible for real-world deployment.

To deploy S-BGP, we must consider the memory overheads for all necessary data structures. One of them is the storage for certificates. In simulation, we assumed every path authentication scheme uses the same amount of memory for certificates as the common base for comparison. Now, we pay closer attention to this amount. As discussed in [25], the scale of the Internet in 2003 required 75–85MB memory on a BGP speaker to store all necessary public key certificates. Taking this amount into account, we conclude that the overall savings by GAS-V against all memory overheads by S-BGP is 60%.

Kent et al. proposed to let ISPs extract only necessary information from certificates and AAs, and push the data to their routers [23]. Routers do not need to store signatures for certificates and AAs. Such optimization may save 50%–60% of the corresponding memory. We adjust the above calculation accordingly. The resulting overall memory saving is estimated as 67%. Moreover, if we consider the extracted information from certificates, the space impacts of different signature algorithms come from their key sizes. Shown in Table 3, the 1024-bit DSA with 160-bit exponent requires about 408 bytes to store the public key and domain parameters. This number is higher than the key size for BLS or RSA, which suggests that we can expect the overall memory saving to be slightly higher than 67%.

Noted in [25], for S-BGP in 2003, the Adj-RIBs space required for RAs is about 30–35MB per peer, and the total requirement for a speaker with tens of peers may be gigabytes. However, current deployed BGP speakers cannot be configured with more than 128M or 256M of RAM. With aggregated path authentication, we still call for additional RAM on routers, unfortunately. The overall 67% memory savings certainly reduce the gap between the memory demands and the reality[1].

6.2 Switching to ECDSA

Elliptic Curve Cryptography (ECC) is emerging as an attractive public-key cryptosystem. Compared to traditional cryptosystems, ECC offers equivalent security with smaller key sizes, faster computations, lower power consumption, as well as memory and bandwidth savings. ECC has been endorsed by the US government as the next generation cryptography. Aggregate signature techniques discussed in this study are examples of using light-weight ECC techniques to achieve equivalent security.

	RSA (1024-bit)	BLS	DSA (1024-bit)
Key Size (bytes)	135	100	408
Sign (*ms*)	7.8	2.2	3.5
Verify (*ms*)	0.4	8.6	4.5

	ECDSA		
	secp192r1	sect163k1	sect163r2
Key Size (bytes)	180	139	155
Sign (*ms*)	1.0	3.1	3.1
Verify (*ms*)	4.4	8.2	8.7

Table 3: Key size and running times of signature algorithms with equivalent security. The benchmark numbers are obtained from OpenSSL 0.9.8 beta version with ECDSA support on a 1GHz processor. Elliptic curves for ECDSA are recommended by NIST for federal government use. Key size accounts for public key and domain parameters.

ECDSA [45] is the elliptic curve counterpart of the traditional DSA algorithm. It has attracted a lot of attentions recently. Popular cryptographic libraries have ECDSA implementation supported. Some hardware architecture, such as Sizzle [19], even allow efficient SSL handshake on small devices using ECDSA. S-BGP can easily switch to use ECDSA as the digital signature algorithm. We use the latest ECDSA implementation in OpenSSL 0.9.8 version to understand its performance. While, ECDSA is certainly much more efficient than RSA computations, it is unclear if it outperforms DSA, except on the key size criterion. Table 3 demonstrates the performance data of different signature algorithms with equivalent security. Compared to DSA, ECDSA signing may perform faster. However,since the majority of cryptographic operations for S-BGP RAs are verifications, we expect no significant improvement on processing latencies. Moreover, both ECDSA using 160-bit curves and DSA produce 40-byte signatures. Certainly, more developments on speeding up ECDSA in the future may lead us to re-examine this issue.

6.3 Hardware Acceleration

Another interesting issue brought up by aggregated path authentication schemes is the hardware acceleration for pairing calculation. Recent rapid developments in improving pairing calculation are driven by applicability to many new EC-based cryptosystems and protocols. To give a few examples, pairing-based systems can support identity-based encryption systems, efficient key agreement protocols, credentials and secret handshakes, provable secure signatures, short signatures, group signatures, blind signatures, proxy signatures, multi-signatures, threshold signatures, intrusion-resilient encryption systems, etc. Many studies have designed and prototyped significantly improved efficient software/hardware implementations of pairing calculation that make these cryptosystems practical. For instance, in 2001 when short signature was first invented, the authors report 2.9 seconds for verification [6]. Then in 2004, the reported running time for BLS verification was reduced to 45.2 *ms* [2]. Now, hardware parallelization allows the pairing calculation to be accomplished within 1.8 *ms* [26]. Our experiments confirm that such hardware implementation allow us to achieve efficient BGP security on both speed and space.

[1]We do not exclude the possibility of adding more memory on routers. If one cannot add memory to a router because the vendor provided no additional slots for more DIMMs, then any scheme that exceeds the memory limitations of deployed routers will be decried as not deployable

In addition, the aggregate verification algorithm in the general aggregate signature scheme provides us even more opportunities for potential hardware parallelization. As discussed in Section 4, the core calculation for aggregate verification is to test whether the following equation holds: $e(g_1, \sigma) = \prod_{i=1}^{k} e(v_i, h_i)$. The pairing calculations on the right hand side are all independent. For a typical small value of k, it should be possible to design a hardware implementation that further parallelizes the calculations. In fact, k is fairly small in the real world. The current BGP CIDR report shows that AS paths are of length 3.7 on average and 11 maximum [8].

Moreover, we envision wide deployment of such hardware accelerator for cryptographic calculations not only for its efficiency, but also for its practicality and usability in the real world. (If student wants to do a doctoral thesis here, please apply to the authors.)

7. CONCLUSION

The Secure BGP proposal has been around for some time. It provides comprehensive security countermeasures to authenticate routing information propagated by BGP speakers. However, route attestations by S-BGP are expensive in terms of processing overhead and space consumption.

We combine the efforts by signature amortization and aggregate signature scheme and design new aggregated path authentication schemes. Choosing various options for each technique, we try out six different constructions for aggregated path authentication (APA)—GAS-V(SW), GAS-T(SW), GAS-V(HW), GAS-T(HW), SAS-V, and SAS-T.

We use simulation to evaluation performance of each construction and compare them with S-BGP and S-A. Experiment results show that GAS-V using hardware implementation of pairing calculation delivers best performance. It has minimum impact on BGP convergence and can substantially reduce 66% of the message length and 60–67% memory cost by S-BGP.

Our work is the first published report applying aggregate signatures to BGP path authentication and analyzing the practicability and performance issues. Our further analysis on real-world deployment and hardware acceleration convince us that GAS-V is an efficient and practical solution for BGP security.

The simulation methodology we apply in this study has limitations. So far, we use simulated network traffic in the experiments. One may suggest use real BGP traffic to drive simulation. This method may produce more realistic analysis results on message size and memory cost. On the other hand, we need a single event to measure convergence time. It is difficult to separate an event from such continuous Update traffic. In the future, we will consider the possibility of using emulation approach to combine the simulated network with realistic BGP traffic.

When a BGP speaker has high degree of connectivity, there exists a trade-off when explicit inclusion of AS numbers as recipients yields a smaller RA than the bit vector. In the simulated 110-AS topology, the bit vector approach is always better than AS number approach, since BGP speakers only have a few peers. We will consider larger simulation models in future to evaluate this trade-off.

Moreover, we are going to explore the issues on storage space for certificate distribution and practical hardware cryptographic accelerators.

Acknowledgment

The authors are grateful to Steve Kent, Russ Housley, B. J. Premore, Hongsuda Tangmunarunkit, and anonymous reviewers for their valuable suggestions. This research has been supported in part by Sun, Cisco, the Mellon Foundation, NSF (CCR-0209144, EIA-9802068), AT&T/Internet2 and the Office for Domestic Preparedness, Department of Homeland Security (2000-DT-CX-K001). This paper does not necessarily reflect the views of the sponsors.

8. REFERENCES

[1] William Aiello, John Ioannidis, and Patrick McDaniel. Origin Authentication in Interdomain Routing. In *Proceedings of the 10th ACM Conference on Computer and Communications Security*, pages 165–178. ACM Press, October 2003.

[2] Paulo S. L. M. Barreto, Ben Lynn, and Michael Scott. Efficient Implementation of Pairing-Based Cryptosytems. *Journal of Cryptology*, 17(4):321–334, September 2004.

[3] Paulo S.L.M. Berreto. A note on efficient computation of cube roots in characteristic 3. Cryptology ePrint Archive, Report 2004/305. `http://eprint.iacr.org/2004/305`, 2004.

[4] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. A Survey of Two Signature Aggregation Techniques. *RSA CryptoBytes*, 6(2):1–10, 2003.

[5] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. In *Proceedings of Eurocrypt 2003*, number 2656 in LNCS, pages 416–432. Springer-Verlag, 2003.

[6] Dan Boneh, Ben Lynn, and Hovav Shacham. Short Signatures from the Weil Pairing. In *Proceedings of Asiacrypt 2001*, number 2248 in LNCS, pages 514–532. Springer-Verlag, 2001.

[7] Dan Boneh, Ben Lynn, and Hovav Shacham. Short Signature from the Weil Pairing. *Journal of Cryptology*, 17(4):297–319, 2004.

[8] CIDR BGP Reports from AS1221 (Telstra), May 2005. `http://www.cidr-report.org/as1221/`.

[9] I. Duursma and H.-S. Lee. Tate pairing implementation for hyperelliptic curves $y^2 = x^p - x + d$. In *Advances in Cryptology - Asiacrypt 2003*, number 2894 in LNCS, pages 111–123. Springer-Verlag, 2003.

[10] Michalis Faloutsos, Petrof Faloutsos, and Christos Faloutsos. On Power-Law Relationships of the Internet Topology. In *Proceedings of ACM SIGCOMM'99*, pages 251–262. ACM Press, 1999.

[11] G. Frey and H. Ruck. A remark considering m-divisibility in the divisor class group of curves. *Mathematics of Computation*, 62:865–874, 1994.

[12] S. Galbraith, K. Harrison, and D. Soldera. Implementing the Tate pairing. In *Algorithm Number Theory Symposium - ANTS V*, volume 2369 of *LNCS*, pages 324–337. Springer-Verlag, 2002.

[13] Lixin Gao. On Inferring Autonomous System Relationships in the Internet. *IEEE/ACM Transactions on Networking (TON)*, 9(6):733–745, December 2001.

[14] Goeffrey Goodell, William Aiello, Timothy Griffin, John Ioannidis, Patrick McDaniel, and Aviel Rubin. Working around BGP: An Incremental Approach to Improving Security and Accuracy in Interdomain Routing. In *The 10th Annual Network and Distributed System Security Symposium*, San Diego, California, February 2003.

[15] Michael Goodrich. Efficient and Secure Network Routing Algorithms. provisional patent filing, `http://www.cs.jhu.edu/~goodrich/cgc/pubs/rout-ing.pdf`, January 2001.

[16] P. Grabher and D. Page. Hardware Acceleration of the Tate Pairing in Characteristic Three. In *Workshop on Cryptographic Hardware and Embedded Systems 2005 (CHES 2005)*, Edinburgh, Scotland, August 2005.

[17] R. Granger, D. Page, and M. Stam. On Small Characteristic Algebraic Tori in Pairing-Based Cryptography. Cryptology ePrint Archive, Report 2004/132. `http://eprint.iacr.org/2004/132`, 2004.

[18] Timothy G. Griffin and Gordon Wilfong. An Analysis of BGP Convergence Properties. In *Proceedings of SIGCOMM 1999*, pages 277–288, August 1999.

[19] Vipul Gupta, Matthew Millard, Stephen Fung, Yu Zhu, Nils Gura, Hans Eberle, and Sheueling Chang Shantz. Sizzle: A Standards-based end-to-end Security Architecture for the Embedded Internet. In *Third IEEE International Conference on Pervasive Computing and Communications*, March 2005.

[20] R. Housley, W. Polk, W. Ford, and D. Solo. Internet X.509 Public Key Infrastructure Certificate and CRL Profile. RFC3280, `http://www.ietf.org/rfc3280.txt`, April 2002.

[21] Russ Housley. S-BGP memory issues are the obstacle for real-world deployment. Personal communication, April 2005.

[22] Yih-Chun Hu, Adrian Perrig, and Marvin Sirbu. SPV: Secure Path Vector Routing for Securing BGP. In *Proceedings of SIGCOMM 2004*, pages 179–192. ACM Press, August 2004.

[23] Stephen Kent, Charles Lynn, Joanne Mikkelson, and Karen Seo. Secure Border Gateway Protocol (S-BGP) – Real World Performance and Deployment Issues. In *The 7th Annual Network and Distributed System Security Symposium (NDSS'00)*, San Diego, California, February 2000.

[24] Stephen Kent, Charles Lynn, and Karen Seo. Secure Border Gateway Protocol. *IEEE Journal of Selected Areas in Communications*, 18(4):582–592, April 2000.

[25] Steve Kent. Securing the Border Gateway Protocol: A Status Update. In *Seventh IFIP TC-6 TC-11 Conference on Communications and Multimedia Security*, October 2003.

[26] T. Kerins, W. P. Marnane, E. M. Popovici, and P.S.L.M. Barreto. Efficient hardware for the Tate pairing calculation in characteristic three. In *Workshop on Cryptographic Hardware and Embedded Systems 2005 (CHES 2005)*, Edinburgh, Scotland, August 2005.

[27] Craig Labovitz, Abha Ahuja, Abhijit Bose, and Farnam Jahanian. Delayed Internet Routing Convergence. In *Proceedings of SIGCOMM 2000*, pages 175–187, August 2000.

[28] Anna Lysyanskaya, Silvio Micali, Leonid Reyzin, and Hovav Shacham. Sequential Aggregate Signatures from Trapdoor Permutations. In *Eurocrypt 2004*, volume 3027 of *LNCS*, pages 74–90. Springer-Verlag, 2004.

[29] Zhuoqing Morley Mao, Ramesh Govindan, George Varghese, and Randy H. Katz. Route Flap Damping Exacerbates Internet Routing Convergence. In *Proceedings of SIGCOMM 2002*, August 2002.

[30] R. Merkle. Protocols for Public Key Cryptosystems. In *Proc 1980 Symposium on Security and Privacy, IEEE Computer Society*, pages 122–133, April 1980.

[31] Victor S. Miller. The Weil Pairing, and Its Efficient Calculation. *Journal of Cryptology*, 17(4):235–261, September 2004.

[32] Multiprecision Integer and Rational Arithmetic C/C++ Library (MIRACL).

[33] Multi-threaded Routing Toolkit. `http://www.mrtd.net`.

[34] S. Murphy. BGP Security Vulnerabilities Analysis. Internet-Draft `http://www.ietf.org/internet-drafts/draft-murphy-bgp-vuln-01.txt`, October 2004.

[35] David M. Nicol, Sean W. Smith, and Meiyuan Zhao. Evaluation of Efficient Security for BGP Route Announcements using Parallel Simulation. *Simulation Practice and Theory Journal, special issue on Modeling and Simulation of Distributed Systems and Networks*, 12(3–4):187–216, July 2004.

[36] Andy T. Ogielski and James H. Cowie. SSFNet: Scalable Simulation Framework - Network Models. `http://www.ssfnet.org`.

[37] Brian Premore. *An Analysis of Convergence Properties of the Border Gateway Protocol Using Discrete Event Simulation*. PhD thesis, Dartmouth College, June 2003.

[38] Y. Rekhter and T. Li. A Border Gateway Protocol 4 (BGP-4). RFC1771, `http://www.ietf.org/rfc1771.txt`, March 1995.

[39] The Route Views Project. `http://www.antc.uoregon.edu/route-views/`.

[40] M. Scott and Paulo S.L.M. Barreto. Compressed Pairings. In *Advances in Cryptology - CRYPTO'2004*, number 3152 in LNCS, pages 140–156. Springer-Verlag, 2004. Updated version: Cryptology ePrint Archive, Report 2004/032. `http://eprint.iacr.org/2004/032`.

[41] B. Smith and J.J. Garcia-Luna-Aceves. Efficient Security Mechanisms for the Border Gateway Routing Protocol. *Computer Communications (Elsevier)*, 21(3):203–210, 1998.

[42] Lakshminarayanan Subramanian, Volker Roth, Ion Stoica, Scott Shenker, and Randy H. Katz. Listen and Whisper: Security Mechanisms for BGP. In *Proceedings of First Symposium on Networked Systems Design and Implementation (NSDI 2004)*, March 2004.

[43] Tao Wan, Evangelos Kranakis, and P.C. van Oorschot. Pretty Secure BGP (psBGP). In *The 12th Annual Network and Distributed System Security Symposium (NDSS'05)*, San Diego, California, February 2005.

[44] Russ White. Securing BGP Through Secure Origin BGP. *The Internet Protocol Journal*, 6(3):15–22, September 2003.

[45] ANSI X9.63. The Elliptic Curve Digital Signature Algorithm (ECDSA). American Bankers Association, 1999.

[46] Meiyuan Zhao, Sean W. Smith, and David M. Nicol. Evaluating the Performance Impact of PKI on BGP Security. In *4th Annual PKI R&D Workshop*, April 2005.