Securing BGP through Keychain-based Signatures

Heng Yin, Bo Sheng, Haining Wang
College of William and Mary

Jianping Pan
University of Victoria

***Abstract*— As the major component of Internet routing infrastructure, the Border Gateway Protocol (BGP) is vulnerable to malicious attacks. While Secure BGP (S-BGP) provides a comprehensive framework to secure BGP, its high computational cost and low incremental deployment benefits seriously impede its wide usage in practice. Using a lightweight symmetric signature scheme, SPV is much faster than S-BGP. However, the speed boost comes at the price of prohibitively large signatures. Aggregated path authentication reduces the overhead of securing BGP in terms of both time and space, but the speed improvement is still limited by public key computation. In this paper, we propose a simple keychain-based signature scheme called *KC-x*, which has low CPU and memory overheads and provides strong incentive for incremental deployment over the Internet. As a generic framework, KC-x has the flexibility of using different signature algorithms. We implement two realizations of KC-x. One is based on RSA called KC-RSA, and the other is based on Merkle hash tree called KC-MT. After characterizing the overheads of KC-RSA and KC-MT, we evaluate their performance with real BGP workloads. Our experimental results show that KC-RSA is as efficient as SAS-V[1], and KC-MT is even 3-fold faster than SPV with a 40% smaller signature. Through the hybrid deployment of KC-MT and KC-RSA, KC-x can achieve both small signature and high processing rate for BGP speakers.**

I. Introduction

The Internet is a global-scale and decentralized network comprised of numerous smaller inter-connected networks, each of which is an *autonomous system* (AS) under a single administration. The routing process among ASes is called *interdomain routing*. The dominant interdomain routing protocol is the Border Gateway Protocol (BGP), and the current version BGP-4 has been widely used for over a decade [2], [3]. However, due to its initial design for a trusted environment [4], BGP is vulnerable to a variety of malicious attacks. For instance, the communication between BGP peers is subject to wiretapping attacks, and a BGP speaker can be compromised to launch a blackhole attack. These attacks cause transmission of fictitious BGP messages, modification or replay of valid messages, or suppression of valid messages.

Many countermeasures [5], [6], [7], [8], [9], [1] have been proposed for securing BGP. Among them, Secure BGP(S-BGP) [8], [10] is the first comprehensive framework for securing BGP. The S-BGP protocol and its associated architecture are currently under consideration for standardization by the Internet Engineering Task Force (IETF). However, due to its extensive use of certificates and asymmetric cryptography, S-BGP is costly in both computation and storage. Moreover, while S-BGP can be deployed incrementally, it provides little incremental benefits if the deployment is not contiguous. Using an efficient symmetric signature scheme (Merkle hash tree), SPV [7] is far more efficient than S-BGP in processing BGP UPDATE messages and provides stronger benefits for incremental deployment, but at the price of significantly greater storage demands, due to its much larger signature. Seeking for efficiency in both computation and storage, aggregated path authentication [1] has been proposed. Among its software options, the Sequential Aggregated Signature with bit Vector (SAS-V) yields the best performance. The improvement in computation, however, is limited, due to the use of asymmetric cryptography. By exploiting BGP's natural path stability, Butler *et al.* [6] significantly reduced the computational cost of BGP path authentication, but at the expense of higher bandwidth cost. With the reasonable bandwidth cost, its performance improvement is still limited. In general, a viable BGP security scheme faces at least the following three challenges. First, since some BGP routers at certain times have very critical performance demand, it should provide sufficiently high processing speed. Second, with high processing speed, the storage and bandwidth overhead should be affordable. Third, it should provide incremental benefits even when not all routers participate. However, none of the existing countermeasures have addressed all these issues successfully.

TABLE I
Comparison of S-BGP, SAS-V, SPV, and KC-x

	Incremental Benefit	Speed	Memory Usage
S-BGP	Weak	Lowest	$>$ SAS-V
SAS-V	Weak	2X speedup	$=$ KC-RSA
SPV	Strong	13X	largest
KC-RSA	Strong	2X	Smallest
KC-MT	Strong	34X	$<$ SPV
KC-Hybrid	Strong	$\geq$ SPV	$<$ KC-MT

In this paper, we propose a simple keychain-based ASPATH protection scheme, called *KC-x*, for securing BGP. The distinct feature of KC-x is that the keys used for signature generation and verification form a chain by themselves, resulting in a strong tie between signatures. Such a construction provides strong benefits for incremental deployment. It can still provide some security protection even with a sparse deployment, and the protection is strengthened with the deployment of KC-x on more routers. As a generic signature framework, KC-x can be realized using any efficient digital signature algorithm. Multiple realizations can co-exist in a hybrid deployment. Indeed, we build two realizations of KC-x using RSA (KC-RSA) and Merkle hash tree (KC-MT), respectively. On one hand, based on RSA-1024, KC-RSA achieves the same performance as SAS-V. Moreover, KC-RSA achieves aggregated

[1]This is the most efficient software approach in Aggregation Path Authentication [1].

signature without modifying the existing RSA implementation, which is required by SAS-V. On the other hand, KC-MT is much simpler in design and more efficient in both computation (i.e., a factor of 3 faster) and storage (i.e., 40% less) than SPV, because it constructs smaller trees and reuses them over multiple signatures.

After characterizing the overheads of KC-RSA and KC-MT, we evaluate their performance under two types of realistic BGP workloads: normal and pathological. Note that KC-RSA and KC-MT can co-exist in a single BGP router. KC-Hybrid refers to the hybrid deployment of KC-RSA and KC-MT across the Internet, in which KC-MT is primarily used in the BGP routers having critical demand for performance, while KC-RSA plays a major role in the remaining routers. KC-Hybrid can achieve both small signature to save space, and high processing rate for handling a high volume of UPDATE messages. Overall, KC-x provides strong benefits for incremental deployment and satisfactory processing speed with modest storage cost and small bandwidth cost, making it a very promising BGP security mechanism for practical deployment. In comparison with S-BGP, SAS-V, and SPV, we summarize the advantages of KC-x in Table I.

The remainder of this paper is organized as follows. Section II outlines the operation and security requirements of BGP. Section III details the design of KC-x for securing BGP. Section IV investigates two realizations of KC-x. Section V characterizes their computation and memory overheads, and evaluates their performance under the real BGP workloads. Section VI surveys the related work. Finally, the paper concludes with Section VII.

II. Interdomain Routing Security

The primary function of the Border Gateway Protocol (BGP) [2], [3] is to exchange network reachability information among BGP speakers. This information is used to construct an AS connectivity graph, in which interdomain routes are established, routing loops are pruned, and routing policies at the AS level are enforced. Thus, the goal of protecting BGP is to ensure the integrity, authenticity and availability of AS graphs. BGP involves two types of control information exchange: one is between peering speakers, and the other is relayed through a series of intermediate speakers. Protecting the peer exchange is just another variant of protecting data communications between any two endpoints, and existing security measures such as IPsec or SSL/TLS should apply. However, protecting the relayed routing exchange among BGP speakers is much more challenging: routing information is transformed when it is propagated through intermediate speakers, some of which may be misconfigured or even compromised. Therefore, BGP routing messages are vulnerable to a variety of malicious attacks, which can result in the injection of false routing messages and the suppression of valid ones.

Researchers have studied possible attacks on BGP [11], [12], [4]. In these studies, attacks are typically classified as passive and active. In passive attacks, attackers simply eavesdrop information off the network. Confidentiality is not a major concern for BGP, and if necessary, can be achieved by employing IPsec [13] between peering speakers. Active attacks are more sophisticated as attackers can manipulate routing messages in the network, which include replay attacks, message insertion, deletion and modification, man-in-the-middle attacks, and denial-of-service attacks. To launch an active attack, adversaries may have to access network links or compromise routers.

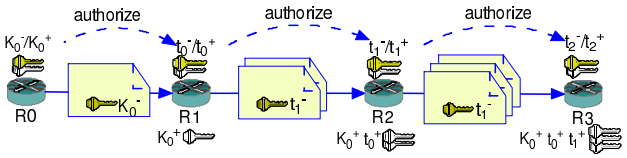


Fig. 1. Overview of keychain-based signature scheme

Therefore, we focus on active attacks, especially the attacks that manipulate BGP UPDATE messages. Other BGP messages such as OPEN, NOTIFICATION, and KEEPALIVE messages can be protected by using IPsec or TCP-MD5 [14] between peering speakers. There are two kinds of falsification attacks on BGP UPDATE messages. One is the network layer reachability information (NLRI)[2] falsification attack, and the other is the ASPATH falsification attack. For example, black holing is one of the severe attacks using falsification, in which adversaries can either advertise a more specific prefix (i.e., NLRI falsification) or shorten ASPATH to "persuade" other routers to prefer the route otherwise not being preferred (i.e., ASPATH falsification).

Unlike blackhole attacks, grayhole attackers may selectively drop traffic flowing through them. Further, in colluding attacks, multiple compromised BGP routers may collude and exchange BGP messages and secrets through a tunnel. These colluding routers can cooperate to launch a blackhole attack and other sophisticated attacks. These advanced attacks cannot be prevented solely by adopting a secure routing protocol, like S-BGP or SPV. Thus, defending against such attacks is beyond the scope of this paper.

This paper investigates an efficient security scheme for protecting ASPATH. Any protection scheme for NLRI is complementary to KC-x, such as PKI-based centralized approach in S-BGP and decentralized one in psBGP.

III. Keychain-based Signature

In this section, we first present the fundamental design of the proposed keychain-based signature scheme, and then discuss its security property. Finally, we describe how to estimate the computation overhead of handling a BGP UPDATE message.

A. Fundamental Design

In KC-x, each BGP speaker (R_i) generates a temporary key pair (t_i^+/t_i^-). As shown in Figure 1, the speaker R_i authorizes its next-hop speaker (R_{i+1}) and passes t_i^- to R_{i+1} in plaintext. The UPDATE message and the temporary public

[2]NLRI refers to the IP prefixes that the UPDATE message and path attributes pertain to.

TABLE II

SUMMARY OF NOTATIONS

N_x	AS number
R_i	a BGP speaker
V_i	a bit vector of BGP speaker R_i
$H(M)$	hash value of the message M
$\{M\}_{K^-}$	the message M signed by using the private key K^-
$K^-(M)$	encrypt the message M by using the private key K^-
$K^+(M)$	decrypt the message M by using the public key K^+

key t_i^+ are signed with the private key (t_{i-1}^-), which is authorized by its preceding speaker (R_{i-1}). In consequence, the above construction forms a chain of authorization. For each BGP speaker, instead of signing an UPDATE message directly with its own private key as S-BGP, the speaker signs the message using a temporary private key that is authorized by its preceding speaker along the ASPATH, and verifies the message by using temporary public keys of all previous speakers along the ASPATH. The only exception is that the speaker R_0 from the origin AS still signs ASPATH with its own private key that is authenticated by PKI, since there is no preceding speaker before R_0. For convenience, a temporary key (private or public) is termed as an *attestation key* in the following discussion.

In more detail, we describe how KC-x protects an UPDATE message from the origin AS N_0, and propagates it to AS N_i. The notations we use in the following discussion are listed in Table II. Here, K_i^+/K_i^- denotes the regular public/private key pair of the BGP speaker of AS N_i, and t_i^+/t_i^- denotes the attestation key pair generated by the speaker of AS N_i.

When the speaker R_0 of AS N_0 advertises some prefixes it owns, it signs the UPDATE message using its own private key K_0^-, which is the same as S-BGP. As proposed in [1], a bit vector denoted as V_0 here is included in the UPDATE message to amortize signature cost. In addition, it carries an attestation key pair t_0^+/t_0^- and signs the attestation public key t_0^+ using K_0^-. To improve performance, this attestation key pair can be generated offline or off-peak in advance, and the next-hop speaker can use it to sign multiple messages. The message that R_0 sends to R_1 is shown as follows:

$$R_0 \rightarrow R_1 : \{N_0; V_0; t_0^+\}_{K_0^-}, t_0^-$$

Upon receiving the above message, R_1 first verifies the signature using R_0's public key K_0^+. If R_1 decides to forward this message to its external peers, R_1 appends its own AS N_1 to the ASPATH. Similarly, R_1 also sends its attestation key pair t_1^+/t_1^- to the next-hop speaker. Meanwhile, R_1 uses R_0's attestation private key to sign the UPDATE message. Like S-BGP, the speaker of KC-x carries all route attestations (RAs) from the received message, and appends its own attestation. However, R_1 needs to remove R_0's attestation private key t_0^-, from the message to maintain the secrecy between R_0 and R_1. The message that R_1 sends to R_2 is shown as follows:

$$\begin{aligned} R_1 \rightarrow R_2 : \quad & \{N_0; V_0; t_0^+\}_{K_0^-} \\ & \{N_0, N_1; V_1; t_1^+\}_{t_0^-}, t_1^- \end{aligned}$$

Generally speaking, the message received by speaker R_i ($0 < i \leq n$) has the following format:

$$\begin{aligned} R_{i-1} \rightarrow R_i : \quad & \{N_0; V_0; t_0^+\}_{K_0^-}, \\ & \{N_0, N_1; V_1; t_1^+\}_{t_0^-} \\ & \ldots \\ & \{N_0, ..., N_{i-1}; V_{i-1}; t_{i-1}^+\}_{t_{i-2}^-}, t_{i-1}^- \end{aligned}$$

Upon receiving the above message, R_i first verifies the received RAs sequentially in the order of that they are signed—one RA is verified using the attestation public key included in the preceding RA. When forwarding the UPDATE message, R_i appends its AS N_i to the ASPATH, including the next-hop AS and its attestation public key t_i^+ in the RA, and signs the RA using R_{i-1}'s attestation private key t_{i-1}^-.

Note that the attestation keys are sent in every UPDATE message. This design simplifies the handling of the cases in which some next-hop speakers are down and then up, because these speakers can always receive the refreshed attestation keys from the latest UPDATE message. A straightforward optimization could be sending attestation key pairs only if necessary to save bandwidth consumption. Since the extra bandwidth consumed by attestation keys is small, we do not include such an optimization in the paper.

B. Integration with BGP

Like the other secure BGP routing protocols, such as S-BGP and SPV, the authentication information in KC-x is transmitted in an optional transitive path attribute in the UPDATE message.

KC-x employs the similar approaches as S-BGP to handle route aggregation and expiration. When generating an aggregated route from several individual routes, a KC-x speaker needs to attach the authentication of all individual routes to the aggregated one. Route announcements and withdrawals are vulnerable to replay attacks, in which a BGP speaker replays a previously-heard UPDATE message. To defend against replay attacks, KC-x incorporates into the signature an expiration date, after which the corresponding route is no longer valid. When the route is about to expire, the original speaker must re-announce this route with a new signature.

C. Security Analysis

Here we discuss the security property of KC-x in two scenarios: full deployment and partial deployment.

When KC-x is fully deployed over the Internet, ASPATH falsification is infeasible. Recall that KC-x incorporates bit vectors to reduce sign operations. For bit vectors to work properly, each BGP speaker pre-establishes an ordered list of its next-hop speakers and distributes the neighbor list to the other speakers via the speaker's X.509 certificate. Even if an adversary steals one or more attestation private keys, she cannot forge ASPATH with any adverse effect. The forged ASPATH has to be "valid" in the sense that for any consecutive two ASes in it, the latter speaker must be in the former's neighbor list.

$$\ldots N_{i-1}, X_{(i-1,1)}, X_{(i-1,2)}, \ldots N_i, X_{(i,1)}, X_{(i,2)}, \ldots, X_{(i,m)}, M$$

When attestation private keys are secure, falsification is still possible in a very limited way. Consider the above generalized ASPATH, in which N_j is KC-x capable speaker, $X_{(i,j)}$ is legacy, and M is malicious. Since M does not know the attestation private key of N_{i-1}, she cannot modify and remove any AS number before and including N_i. However, she can arbitrarily modify the portion from $X_{(i,1)}$ to $X_{(i,m)}$ inclusively, as long as the AS number following N_i satisfies N_i's bit vector. Generally speaking, the room for the adversary to forge an ASPATH is between the preceding KC-x capable AS to itself. With the increased deployment of KC-x capable speakers, such a security hole becomes smaller.

Hence, KC-x does not need to be deployed contiguously, and is incrementally deployable.

D. Computational Overhead

We now estimate the computation overhead of processing UPDATE messages in KC-x. When receiving an incoming BGP UPDATE message, a BGP speaker needs to validate this message. That is, the speaker checks the authenticity of NLRI and ASPATH. The authenticity of NLRI is validated by matching with the cached AA (Address Attestation) information without cryptographic computation. The validation of ASPATH involves a certain number of signature verifications, which depend on the number of ASes in the ASPATH. Note that the first signature is different from the following signatures, because it is generated using the originating speaker's private key K_0^-. In the implementation, K_0^+/K_0^- is an RSA key pair, and the verification of the first signature is lightweight, as we will show in Section IV-A. Furthermore, to avoid unnecessary overhead, KC-x only validates those UPDATE messages that cause routing table changes, as S-BGP does [15].

When propagating an UPDATE message to the next-hop speakers, the BGP speaker appends its own AS number into the ASPATH and signs the UPDATE message. Due to the use of a bit vector, only one signing operation is required.

When signing an UPDATE message, the speaker needs to attach an attestation key pair for the next-hop speakers. Since the generation of attestation key pairs can be performed offline in advance, its computation overhead is negligible in processing UPDATE messages. Therefore, the estimated computation overhead of handling an outgoing UPDATE message is given below:

$$C \approx Length(\texttt{ASPATH}) \times Time(verify) + Time(sign) \quad (1)$$

From Equation 1, we can see that the computation cost depends only on how fast a signature algorithm can sign and verify a signature, and how long an ASPATH is.

IV. Two Realizations

In this section, we investigate the use of two signature algorithms, RSA and Merkle hash tree, to realize the keychain-based scheme. For convenience, we call these two realizations KC-RSA and KC-MT, respectively.

A. KC-RSA

S-BGP chooses DSA rather than RSA to protect UPDATE messages due mainly to its smaller signature. Specifically, RSA-1024 yields a 128-byte signature, whereas DSA yields only a 40-byte signature. In addition, DSA supports pre-computation, which greatly reduces the cost of signing operation.

On the other hand, while RSA is about 2-fold slower than DSA in signing with the same key length, it is one order of magnitude faster in verifying. In a simple experiment, in which we use a Pentium-4 1.8GHz CPU, RSA-1024 takes 6.8ms in signing but only 0.35ms in verifying. By contrast, DSA takes 3.8ms (0.03ms with pre-computation) in signing, but 4.6ms in verifying. Assuming that each route contains an average of 3.7 ASes [16], RSA-1024 is still 2-fold faster than DSA-1024 even when pre-computation for DSA is enabled.

1) Signature aggregation: In [1], Zhao, et al. proposed two schemes, general aggregate signature and sequential aggregate signature, to aggregate multiple signatures into one signature. The signature aggregation reduces the length of UPDATE messages and memory consumption of routing tables. Although these two schemes can also be applied to KC-RSA, they require the modification of the existing RSA signature algorithm. In this paper, we propose a new scheme of signature aggregation that eliminates this requirement. The proposed scheme leverages the exclusive-OR operation and is termed as *XOR-ed aggregate signature*.

We still use the previous example to illustrate how XOR-ed aggregate signature works. A new symbol S_i denotes the signature generated by R_i. There is no change for R_0, which sends S_0 and its attestation private key t_0^- as follows.

$$S_0 = K_0^- \left(H(N_0; V; t_0^+)\right). \quad (2)$$

For each following R_i, instead of appending S_i to the existing sequence of signatures $S_0, S_1, ..., S_{i-1}$, R_i incorporates the preceding signature S_{i-1} into its own signature S_i, which is shown as follows:

$$S_i = t_{i-1}^- \left(S_{i-1} \oplus H(N_0, ..., N_i; V; t_i^+)\right) \quad (3)$$

R_{i+1} verifies S_i in the following steps: (1) decrypt S_i: $t_{i-1}^+(S_i) = S_{i-1} \oplus H(N_0, ..., N_i; V; t_i^+)$; (2) compute the hash value of the message: $h = H(N_0, ..., N_i; V; t_i^+)$; (3) Recover S_{i-1}: $t_{i-1}^+(S_i) \oplus h = S_{i-1} \oplus H(N_0, ..., N_i; V; t_i^+) \oplus H(N_0, ..., N_i; V; t_i^+) = S_{i-1}$; (4) $i = i-1$, if $i \neq 0$, go to step (1); (5) use R_0's public key K_0^+ to verify S_0. All the signatures are correct only if S_0 passes the verification. Since exclusive-OR is very lightweight, the cost of additional exclusive-OR operations induced by XOR-ed aggregate signature is negligible.

Note that attestation public keys are carried along with the aggregate signature in the UPDATE messages. Each public key is also 128-byte long for RSA-1024. An optimization for memory consumption in the RIB is to store public keys in

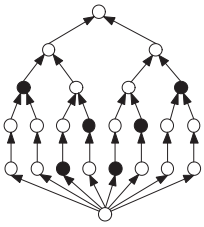


Fig. 2. **An example of HORS signature**. A hash tree with 8 leaf nodes is derived from a secret value, and two leaf nodes are disclosed for each signature. The gray circles denotes the disclosed values in a signature. Using these values, a verifier can recalculate the root value.

a different database from RIB, since one public key can be reused among multiple routes.

B. KC-MT

Invented by Ralph Merkle, a Merkle hash tree [17] creates secure signatures based on hash functions (e.g. SHA-1). Due to the use of lightweight hash functions, Merkle-tree-based signature scheme is far more efficient than those based on asymmetric crypto-systems. The basic Merkle-tree-based signature works as follows: Alice populates n leaf nodes from a secret and builds a hash tree; when signing a message, Alice hashes the message and maps it into a leaf node, and then discloses the leaf node accompanying with some corresponding intermediate nodes to Bob; after receiving the leaf node and the intermediate nodes, Bob can compute the root value, and compare it with the value given by Alice. Since the chance of a forged message mapping to the same leaf node as a legitimate message is $\frac{1}{n}$, we need a large hash tree to achieve high security.

To reduce tree size, a few variants have been proposed. The HORS signature [18] is one of them, which can achieve fast verification with a smaller hash tree. Instead of disclosing one leaf node, the HORS signature discloses m leaf nodes for each signature. There is only a chance of $1/\binom{n}{m}$ that two messages yield the same signature. Figure 2 shows an example of $(n = 8, m = 2)$.

SPV uses Merkle hash tree and HORS signature to secure BGP routing [7]. In SPV, an origin BGP speaker constructs a single large hash tree in three hierarchies. In the lowest hierarchy, a subtree with 256 leaf nodes is used to authenticate a single AS in the ASPATH. Then, all root values of these subtrees form a tree in the second hierarchy, and its root value, named as epoch public key, is used to authenticate one ASPATH protector. Finally, multiple epoch public keys form a tree in the highest hierarchy, whose root value called multi-epoch public key authenticates these epoch public keys. The origin AS signs this multi-epoch public key, and disseminates the multi-epoch public key certificate to BGP speakers.

KC-MT also makes use of HORS signatures in conjunction with a hash tree. Unlike SPV, KC-MT has very simple constructions. For a Merkle hash tree of KC-MT, the secret used to populate all leaf nodes is treated as a private key, and the root value of the tree is a public key. According to the fundamental design of KC-x, a speaker R_i generates an attestation key pair by building a Merkle hash tree based on a secret t_i^- (i.e. attestation private key) and calculates the root value t_i^+ (i.e. attestation public key). It then signs the ASPATH and t_i^+ using the hash tree derived from the secret t_{i-1}^- given by its preceding speaker R_{i-1}, and then forwards the signed message and t_i^- to the next-hop speaker.

For a Merkle hash tree, security degrades when more signatures are generated, because more leaf nodes are disclosed. SPV chooses $(n = 256, m = 6)$ to ensure the forgery probability of 2^{-11} after 15 signatures. Similarly in KC-MT, we limit the maximum number (Σ) of signatures a single tree can yield. We will discuss the selection of n and m and the corresponding maximum number of signatures in Section V-A.1. To ensure that one tree signs at most Σ messages, the issuing speaker generates a new attestation key pair after sending Σ UPDATE messages.

1) Comparison between KC-MT and SPV: Based on a different security construction, KC-MT is simpler in nature than SPV in three aspects. First, in SPV, the concept of epoch is introduced to thwart repeatable and predictable fraud. Adding an epoch number to each hash operation makes the probability of forgery independent between epochs. The cost is that one more hierarchy has to be constructed in the Merkle tree, more values are included into the signature, and more hash operations are required to authenticate an ASPATH. In contrast, KC-MT prevents this attack implicitly by its inherent rekeying mechanism. That is, in KC-MT, each speaker independently builds a tree, authorizes its next-hop speaker to use it, and periodically reconstructs the tree.

In addition, in SPV, the multi-epoch public key has to be distributed, in the form of a certificate signed with the origin BGP speaker's private key, to all speakers who need to verify an ASPATH originated from the issuer. SPV needs a certificate distribution mechanism, either fulfilled by a separate protocol or conveying certificates within UPDATE messages by the means of erasure codes. In contrast, KC-MT does not require any extra mechanisms for distributing certificates, since public keys are distributed via UPDATE messages.

Finally, the design of SPV is vulnerable to multi-path truncation attack, in which a single-ASN private key obtained from a shorter ASPATH can be used to truncate a longer ASPATH from the same origin. To counter such an attack, SPV introduces an additional level to the ASPATH authenticator, and degrades private values to semi-private values gradually along the path. Obviously, this induces design complexity and extra performance overhead to SPV. In contrast, KC-MT is not vulnerable to this kind of attack, and no additional protection is needed.

The design simplicity of KC-MT brings performance benefits. The costs of both verifying and signing are reduced compared to SPV. The overhead reduction of verifying lies in fewer hash operations: for each AS in the ASPATH, KC-MT needs only to compute the root value of each small tree, whereas SPV needs more hash operations to compute the multi-epoch public key from the single-ASN public keys and several other intermediate values. The cost reduction of

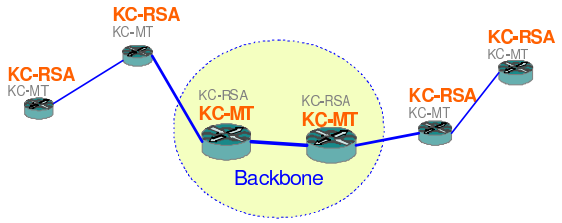


Fig. 3. **Hybrid deployment on BGP speakers in the Internet.** The BGP routers serving high volume of BGP messages use KC-MT to sign UPDATE messages, while the other BGP routers use KC-RSA. By exploiting benefits of these two algorithms, the hybrid deployment can achieve faster speed in processing UPDATE message and smaller footprint in storage and transmission.

signing is achieved by caching the intermediate values during the first signing operation, and then the number of hash operations for the subsequent signing operations on the same tree can be greatly reduced. The number of cached trees can be determined by AS degree. As shown in [19], AS degree follows a power law distribution, and 90% of ASes have less than 10 neighboring ASes. Even for the ASes (with a probability of 0.01%) having over 1000 neighboring ASes, only about 1MB buffer is required to cache all trees with 512 leaf nodes. By contrast, the Merkle tree in SPV is substantially larger, and different origin BGP speakers use different trees. Thus, it would be unaffordable for SPV to cache all trees. Note that as mentioned in Section III, attestation key pairs for next-hop speakers can be generated offline. Even for online computation, its small cost is further amortized over Σ outgoing UPDATE messages. So, the cost of key generation is negligible.

C. Hybrid Deployment

KC-MT is very fast but has relatively large signatures, whereas KC-RSA yields very small signature but is slower than KC-MT. Here we present a hybrid deployment approach to achieving both small signature and high processing rate.

In the hybrid deployment, both KC-RSA and KC-MT are installed on a BGP router, and hence the router knows how to verify signatures of both KC-RSA and KC-MT. In addition, it chooses either KC-RSA or KC-MT to be the primary for signing UPDATE messages. The routers requiring high processing rate can choose KC-MT to be the primary, while the other routers may choose KC-RSA to sign messages. Figure 3 illustrates such a layout.

For the hybrid deployment to work properly, when a speaker chooses one algorithm as the primary, its preceding speaker has to issue the corresponding attestation key pair for that algorithm. Since the amortized cost of key generation for either KC-RSA or KC-MT is negligible, it is feasible for the preceding speaker to issue both kinds of key pairs and let the receiver choose which one to use. This greatly simplifies the hybrid deployment over the Internet.

As we will show in Section V, the speed of verifying a KC-RSA signature is in the same order of magnitude as that of KC-MT. Thus, with the hybrid deployment, the performance of handling an UPDATE message on a router with KC-MT as the primary will be comparable to that of pure KC-MT deployment. The benefit, however, is much smaller footprint of signatures, due to the signature aggregation in KC-RSA.

V. Evaluation

In this section, we quantify the computation and memory overheads of the keychain-based schemes, which include KC-RSA, KC-MT, and the hybrid of these two, and compare them with those of S-BGP, SAS-V, and SPV under real traces.

A. CPU Overhead

To accurately assess the computation overheads of these different schemes, we implement the HORS signature algorithm using AES [20], [21], and conduct a series of experiments on a modest PC with a Pentium-4 1.8GHZ CPU. We characterize the computation overhead of each individual operation as signing, verification, and key generation. Then, we evaluate and compare the BGP performance with different protection mechanisms under two kinds of workloads: normal and pathological. Before presenting the experiment results of overhead estimation, we first detail the parameter setting of Merkle tree in KC-MT. This is because the operation overhead of KC-MT is highly dependent upon the parameter setting of the Merkle tree.

1) Merkle-tree Configuration: To achieve a forgery probability around $p = 2^{-11}$ after yielding 15 signatures, SPV selects $n = 256$ and $m = 6$ [7]. For KC-MT, by caching the intermediate values of the first signing, we can significantly reduce the overhead of subsequent signing operations. Therefore, KC-MT may choose different (n, m) to run even faster and yield smaller signature without degrading its security, i.e., achieving the same level of security as SPV. With different selections of (n, m), the experiment results of KC-MT are shown in Figure 4.

There are five factors affecting the CPU overhead of KC-MT, including initial signing ($S1$), subsequent signing ($S2$), verification (V), and the maximal number of signatures (Σ). The average signing cost S is:

$$S = S1/\Sigma + (1 - 1/\Sigma) \times S2 \tag{4}$$

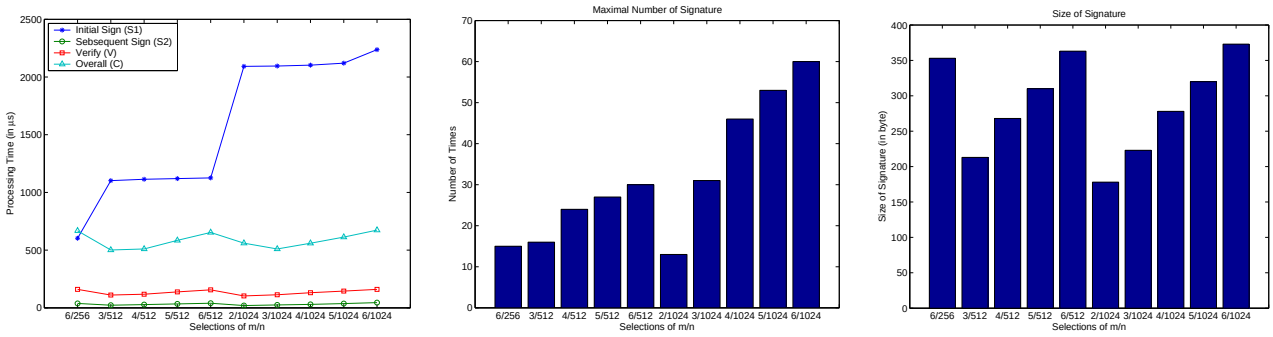
Since the cost of key generation is amortized over Σ messages, according to Equations 1 and 4, the overall computation cost of handling one UPDATE message is:

$$C = Length(ASPATH) \times V + S1/\Sigma + (1 - 1/\Sigma) \times S2 \tag{5}$$

Figure 4(a) shows the overhead breakdown of the first three factors and the overall cost, while Figure 4(b) shows Σ the maximal number of signatures, which a Merkle tree of (n, m) needs to issue, to ensure the same forgery probability (around $p = 2^{-11}$) as SPV[3].

The overhead of initial signing is proportional to n, since most of the internal values of the tree need to be computed

[3]Here we ensure the same forgery probability as SPV just for fair comparison. With different parameters, KC-MT can definitely provide higher-level security protection



(a) (b) (c)

Fig. 4. Operation cost of KC-MT with different (n, m)

TABLE III
SPEED OF INDIVIDUAL OPERATIONS (IN μs)

	S-BGP	SPV	SAS-V/KC-RSA	KC-MT
Sign	3,802/30	703	6,800	90
Verify	4,607	191	350	111

TABLE IV
CHARACTERISTICS OF TRAFFIC TRACES

	Normal	Pathological
Duration(s)	871	876
Total Announcements	1121	88777
Average Rate(/s)	1.29	101.34
Maximum Rate(/s)	117	6764

for the intermediate values required by the signature. For the subsequent signings, as the internal values have been cached, its computational overhead is substantially decreased (less than $50\mu s$). Thus, considering the number of signing operations on the same Merkle tree, we compute the average overhead of signing, which is as low as $73\mu s$ when $n = 512$ and $m = 4$.

As shown in Figure 4(a), the overhead of verification is between $103\mu s$ for $(1024, 2)$ and $197\mu s$ for $(1024, 6)$. Note that the total CPU overhead of KC-MT for handling an UPDATE message is under the assumption that the average length of ASPATH is 3.7. In our experiments, the parameter setting (n, m) with the lowest total CPU overhead is $(512, 3)$. In addition to processing speed, we also need to consider memory consumption of KC-MT and balance these two metrics. Figure 4(c) shows the dynamics of signature size with different Merkle-tree configurations, in which a Merkle tree of $(512, 3)$ yields the second smallest signature. Hence, we set KC-MT as $n = 512$ and $m = 3$.

2) Individual Operation Cost: The individual overheads of signing, verification, and key generation, with respect to S-BGP, SAS-V, SPV, KC-RSA, and KC-MT, are listed in Table 3. Without pre-computation in DSA, the signing cost of S-BGP is 3,802μs. However, it is only 30μs with pre-computation enabled. To be fair in comparison, the signature amortization based on the bit vector is assumed to be available for all candidates, since it is compatible with all of them and easy to implement. We also assume pre-computation in DSA is enabled. To simplify evaluation, the signature cache is not considered, which is also fair since it causes the same effect to all of them.

Based on the above assumptions, we apply Equation 1 to assess the performance of all schemes. Here we still assume that the average ASPATH length is 3.7. We observe that S-BGP is the slowest. SAS-V and KC-RSA are a factor of 2.2 faster than S-BGP, while SPV and KC-MT are 12 and 34 fold faster than S-BGP, respectively. Compared to SPV, KC-MT is about 3 fold faster. In the hybrid deployment, for a router with KC-MT being its primary, even in the worst case that all signatures in ASPATH are of KC-RSA, it is 12.3 fold faster than S-BGP, and still slightly faster than SPV.

3) Performance under Real Workloads: With the knowledge of individual operation cost for S-BGP, SPV, SAS-V, KC-RSA, and KC-MT, we further evaluate the overall performance of securing UPDATE messages at a BGP router under real workloads. Besides normal workloads, we are particularly interested in the scenario of a backbone BGP router experiencing an abnormally high workload due to the outbreak of malicious attacks. One important metric is the delay of handling an UPDATE message at a BGP router, which includes the processing time and the waiting time. The delay is crucial to the convergence speed of Inter-domain routing over the Internet.

The workloads are obtained from UPDATE message traces from *routeviews* [22], in which the *routeviews* routers record the outgoing UPDATE messages of the observed ASes they peer with. From the knowledge of the workloads and the cryptographic operation cost, we can infer the distribution of delayed outgoing UPDATE messages caused by each BGP protection scheme.

We choose two typical UPDATE message traces to represent normal and pathological workloads, respectively. For normal workload, we select the trace of a router (AS 3277) on Jun 1st, 2005. This trace represents the average traffic load among most available BGP routers. For pathological workload, we choose the trace of a router (AS 7911) on Jan 24th, 2003, when SQL worms were spreading quickly across the Internet. The chosen trace has the maximum number of UPDATE messages among all observed BGP routers during that period. The statistics of these two traces are listed in Table IV.

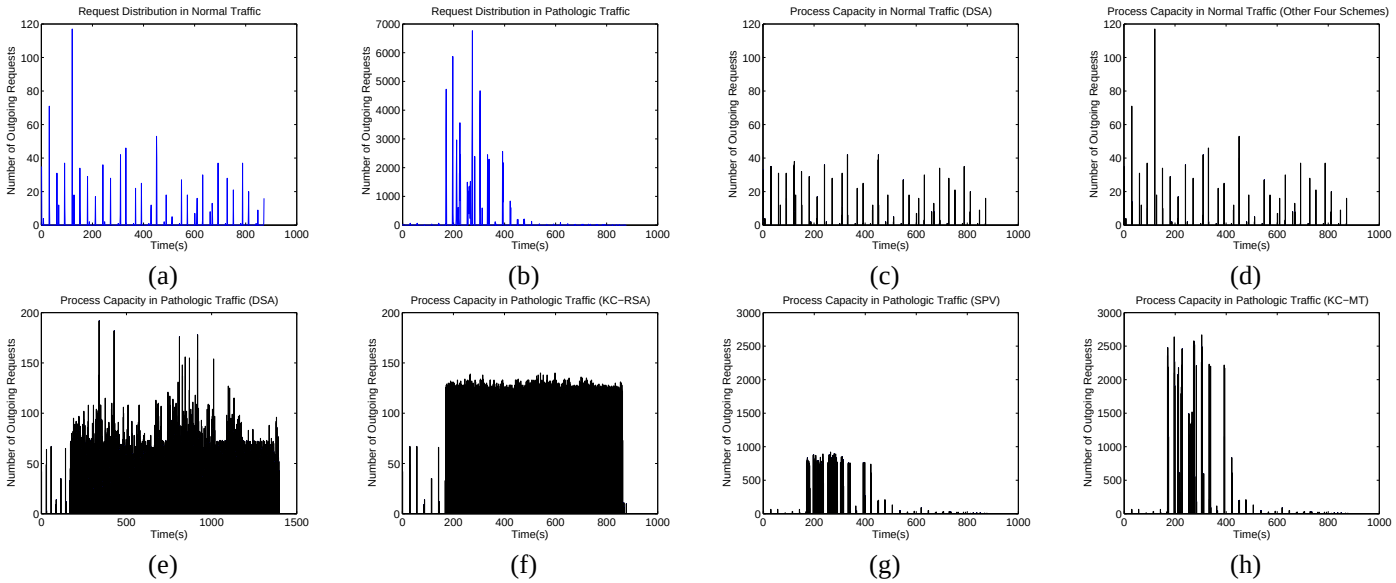


Fig. 5. **Outgoing UPDATE message rates under normal and pathological conditions with or without secure protections:** (a) normal traffic with no protection; (b) pathological traffic with no protection; (c) normal traffic with S-BGP; (d) normal traffic with SAS-V, SPV,KC-RSA, and KC-MT; (e) pathological traffic with S-BGP; (f) pathological traffic with KC-RSA and SAS-V; (g) pathological traffic with SPV; (h) pathological traffic with KC-MT.

The distributions of outgoing UPDATE messages are extracted from these two traces and shown in Figures 5(a) and (b). We observe that the UPDATE messages are sent in burst even under normal condition, because of the rate limiting mechanism in BGP [3].

Figures 5(a) and (b) illustrate the distributions of delayed UPDATE messages under the normal condition caused by different secure schemes. Since the traffic load is relatively light, SPV, SAS-V, and our keychain-based schemes (KC-RSA and KC-MT) can handle all UPDATE messages within one second. As shown in Figure 5(d), the distribution of outgoing messages exactly match the one in Figure 5(a), due to the resolution of one second. For S-BGP, some peak bursty traffic exceeds the processing capacity of DSA, leading to noticeable delay as shown in Figure 5(c). With respect to different BGP secure mechanisms, we summarize their average and maximum delay times under normal condition in the second column of Table V. In the average cases, all schemes can handle an UPDATE message within one second. However, in the worst cases, the delay time of S-BGP increases to 3.633s while the others are still less than one second.

Figures 5(e)-(h) show the results of different secure schemes under the pathological condition. As shown in Table IV and Figure 5(b), the average message arrival rate of the pathological traffic is 2-order of magnitude larger than that of the normal traffic, and the peak has 4000∼7000 arrivals per second and lasts for 3 to 4 minutes. Under this substantially heavy workload, all five secure schemes reach their full processing capacity. As Figure 5(e) shows, on average S-BGP can only process about 60 messages per second, while SAS-V and KC-RSA can handle 120 messages per second, shown in Figure 5(f). Such low processing capacities of these three schemes cause unacceptably large delays up to 21 minutes. Using the efficient HORS signature based on Merkle tree, SPV, as shown in Figures 5(g), can process about 800 messages per second, while KC-MT, shown in (h), is able to process around 2,500 messages per second. Both schemes have some idle time during the peak period, and they can digest the bursty traffic within a few seconds. Due to the accumulation effect of highly bursty traffic under the pathological workload, the average and maximum delays of KC-MT are a factor of 5 less than those of SPV. In the third column of Table V, we summarize the average and maximum delays of different secure schemes under the pathological condition. In the hybrid deployment of KC-x, the performance of a router with KC-MT as primary is between SPV and KC-MT.

TABLE V
DELAY IN NORMAL/PATHOLOGICAL TRAFFIC (IN *s*)

	Normal (avg/max)	Pathological (avg/max)
S-BGP	0.600 / 3.201	515.842 / 952.834
SPV	0.038 / 0.215	4.163 / 14.09
KC-RSA	0.183 / 1.038	251.9 / 454.3
KC-MT	0.015 / 0.088	0.809 / 3.224

B. Memory Consumption

In practice, BGP routers have limited memory space (e.g., 256M). A BGP security mechanism that yields relatively large signatures could easily exceed the memory limit of many BGP routers, impeding its wide deployment. Therefore, memory consumption is another important metric for evaluating BGP routing security mechanisms. Since the size of signature is the dominant factor in determining the memory consumption, we focus on the memory requirement of signatures.

To estimate the memory consumption in real scenarios, we select several typical BGP routers from [23]. Then, we compute the memory consumption of signatures in different BGP security schemes under these real routers' working condition. The results are listed in Table VI. Among these four schemes,

TABLE VI
MEMORY CONSUMPTION OF SIGNATURES (IN MB)

ASN	RIB entries	ASPATH	S-BGP	SPV	KC-RSA & SAS-V	KC-MT	Hybrid 1:1 (2:1)
1221	211721	3.555	28.7	253	25.8	152.5	102.1 (76.7)
4637	163918	3.356	21	185	20	111.5	75.7 (57.2)
7660	167288	4.46	28.5	250.8	20.4	151.2	96 (70.8)

KC-RSA and SAS-V are the most economical. Independent of the length of ASPATH, the size of its signature is only 128 bytes. In all three selected BGP routers, the memory consumption of KC-RSA is less than 26MB. Using DSA, S-BGP yields 40-byte signature for each AS in the ASPATH. Thus, S-BGP consumes slightly more memory than KC-RSA, depending on the average length of ASPATH. SPV and KC-MT also yield one signature for each AS, but signature size may vary with different message content. In our implementation, on average, the Merkle tree of (256,6) selected by SPV yields 353-byte signatures, while the tree of (512,3) selected by KC-MT yields 213-byte signatures. Thus, SPV is the most expensive. It consumes as much as 253MB memory, which may exceed the memory limit of most deployed BGP routers. Due to smaller signatures, the memory consumption of KC-MT is only 60% of that of SPV. As discussed in Section IV-C, the hybrid deployment of KC-RSA and KC-MT can further reduce the total size of signatures. Assume that on ASPATHs, the ratio of signatures of KC-RSA to KC-MT is 1:1 or 2:1. The memory consumption of the hybrid deployment is only about 40% or 30% of that of SPV. The memory requirements of around 90MB and 70MB in these two hybrid deployments are reasonably low and affordable for most of the existing BGP routers.

VI. RELATED WORK

The Secure Border Gateway Protocol (S-BGP) [8], [10] was proposed by BBN to provide strong security for BGP. S-BGP relies on Public Key Infrastructure (PKI) to assign the ownership of an IP prefix to an AS, and to authenticate the identity of a BGP router. To protect the ASPATH from modification and truncation, each *enroute* BGP speaker verifies *route attestations* issued by previous ASes in the ASPATH via their public keys, and when propagating this message, appends its own AS number, and creates its own *route attestation* via its private key. However, due to public key cryptography, S-BGP is expensive in both computation and storage [16], [15], making it inefficient in realistic deployments.

Several efforts have been made to improve the performance of S-BGP. Secure Path Vector (SPV) [7] is designed to use symmetric cryptographic mechanisms to provide integrity protection. While the prefix authentication still relies on PKI, the route attestation is realized by employing a lightweight symmetric one-time signature in conjunction with a Merkle hash tree. The originating speaker builds a large Merkle hash tree, and discloses some nodes of the tree to the following speakers. Each *enroute* speaker can verify the received AS-PATH by using the disclosed values. Additionally, it signs its own AS number into the ASPATH using the subtree derived from the given single-ASN private key. SPV is claimed to be a factor of 22 faster than S-BGP, at the price of significantly high storage demand. Moreover, the fairly complicated design makes it challenging to implement and deploy SPV in practice.

Still using public key cryptography, Zhao et al. [1] applied several optimizations to improve the processing speed and reduce the storage cost of S-BGP, by combining the time-efficient scheme of signature amortization with the space-efficient techniques of aggregate signatures. However, the performance improvement is still limited by the expensive public key computation.

By exploiting the stability of path advertisement, Butler et al. [6] investigated several optimization solutions to amortize cryptographic operations over many verifications. However, the performance improvement is achieved at the expense of higher bandwidth cost. For example, the all path scheme has the biggest processing speed gain (97.3% load reduction), but with prohibitively high bandwidth overhead (as much as 139 megabytes per minute). The origin path scheme, the second fastest, has limited improvement (about 86.3% reduction), with reasonable bandwidth cost. This approach can be applied in conjunction with the other schemes, including KC-x, to further improve the performance.

All the above schemes, including ours, assume a global and public-key infrastructure (PKI). However, building such an infrastructure is challenging. Some efforts have been made to address this issue, and they are complementary to our approach. Pretty Secure BGP (psBGP) [24] represents a new solution for prefix authentication via the construction of a decentralized authentication system, rather than a centralized infrastructure employed by S-BGP. Each AS maintains a *prefix assertion list* (PAL), which includes the address ownership assertions of the local AS and its peers. The prefix information is validated by checking the consistency of PALs of the peers around the origin. Recently, Grassroots-PKI [25] has been proposed as an evolutionary approach to enabling the incremental construction of a global PKI.

Independent of S-BGP, a few completely different protection schemes have been proposed. Secure Origin BGP (soBGP) [26] provides a secure registry mechanism against which a BGP speaker can check the authenticity of an originating AS and the validity of an ASPATH. Interdomain Route Validation (IRV) [27] proposes to setup an IRV server in each AS responsible for validating the route information, and the local IRV server queries other relevant IRV servers for the validity when necessary. “Listen and Whisper” [9] is a lightweight protection with less guarantee. “Listen” detects invalid routes in the data plane by detecting incomplete TCP connections, while “whisper” uncovers invalid route

announcements by detecting inconsistency among multiple UPDATE messages originating from the same AS. Pretty Good BGP [28] is another lightweight protection scheme, Its essence is to detect suspicious advertisements using historical hints, and delay the propagation of them. Suspicious origin ASes are temporarily assigned a low preference, and suspicious sub-prefixes are temporarily ignored. In addition, there is an approach [29] using centralized servers with identity-based cryptography and encrypted search to verify received BGP UPDATE messages.

VII. Conclusion

In this paper, we present KC-x, a keychain-based security mechanism for securing BGP. KC-x builds a chain of key authorization along an ASPATH. Such a key chain creates a strong tie between the BGP speakers along the ASPATH. Hence, KC-x provides strong incremental benefits for partially deployment over the Internet. Moreover, as a generic security mechanism, KC-x can be realized using any efficient digital signature algorithm, and support the hybrid deployment. To demonstrate this approach, we investigate and evaluate two realizations: KC-RSA and KC-MT. We believe that the inherent simple design, easy implementation, strong benefits for incremental deployment, high processing speed, and modest memory usage will make KC-x a very promising BGP security mechanism for wide deployment over the Internet.

References

[1] Meiyuan Zhao, Sean W. Smith, and David M. Nicol, "Aggregated path authentication for efficient BGP security," in *Proceedings of the 12th ACM Conference on Computer and Communication Security (CCS 2005)*, November 2005.

[2] Y. Rekhter and P. Gross, "Application of the Border Gateway Protocol in the Internet," *Request for Comments: 1772, Internet Engineering Task Force*, March 1995.

[3] Y. Rekhter and T. Li, "A Border Gateway Protocol 4 (BGP-4)," *Request for Comments: 1771, Internet Engineering Task Force*, March 1995.

[4] Sandra Murphy, "BGP security vulnerabilitis analysis," *Internet draft, Internet Engineering Task Force*, October 2004.

[5] William Aiello, John Ioannidis, and Patrick McDaniel, "Origin authentication in interdomain routing," in *Proceedings of the 10th ACM conference on Computer and Communication Security (CCS 2003)*. 2003, pp. 165–178, ACM Press.

[6] Kevin Butler and William Aiello, "Optimizing bgp security by exploiting path stability," in *Proceedings of the 13th ACM conference on Computer and communications security (CCS 2006)*, November 2006.

[7] Yih-Chun Hu, Adrian Perrig, and Marvin Sirbu, "SPV: Secure path vector routing for securing BGP," in *Proceedings of ACM SIGCOMM 2004*, September 2004.

[8] Stephen Kent, Charles Lynn, and Karen Seo, "Secure border gateway protocol (S-BGP)," *IEEE JSAC on Network Security*, vol. 18, no. 4, pp. 582–592, April 2000.

[9] L. Subramanian, V. Roth, I. Stoica, S. Shenker, and R. Katz, "Listen and whisper: Security mechanisms for BGP," in *Proceedings of First Symposium on Networked Systems Design and Implementation (NSDI 2004)*, March 2004.

[10] Charles Lynn and Karen Seo, "Secure BGP (S-BGP)," *Internet draft, Internet Engineering Task Force*, June 2003.

[11] A. Barbir, S. Murphy, and Y. Yang, "Generic threats to routing protocols," *Internet draft, Internet Engineering Task Force*, October 2004.

[12] S. Convery, D. Cook, and M. Franz, "An attack tree for the Border Gateway Protocol," *Internet draft, Internet Engineering Task Force*, February 2004.

[13] S. Kent and R. Atkinson, "Security Architecture for the Internet Protocol," *RFC 2401, Internet Engineering Task Force*, November 1998.

[14] A. Heffernan, "Protection of BGP sessions via the TCP MD5 signature option," *RFC 2385*, August 1998.

[15] Stephen Kent, Charles Lynn, Joanne Mikkelson, and Karen Seo, "Secure Boder Gateway Protocol (S-BGP) – real world performance and deployment issues," in *Proceedings of Network and Distributed System Security Symposium (NDSS 2000)*, February 2000.

[16] Stephen Kent, "Securing the Boder Gateway Protocol: A status update," *the seventh IFIP TC-6 TC-11 Conference on Communication and Multimedia Security*, October 2003.

[17] Ralph Merkle, "Protocols for public key crptosystems," *IEEE Symposum on Security and Privacy*, 1980.

[18] Leonid Reyzin and Natan Reyzin, "Better than BiBa: Short one-time signatures with fast signing and verifying," *Information Security and Privacy–7th Australasian Conference (ACSIP 2002)*, July 2002.

[19] Hyunseok Chang, Ramesh Govindan, Sugih Jamin, Scott J. Shenker, and Walter Willinger, "Towards capturing representative AS-level Internet topologies," *SIGMETRICS Performance Evaluation Review*, vol. 30, no. 1, pp. 280–281, 2002.

[20] J. Daemen and V. Rijmen, "AES proposal: Rijndael," March 1999.

[21] S. Matyas, C. Meyer, and J. Oseas, "Generating strong one-way functions with cryptographic algorithm," *IBM Technical Disclosure Bulletin*, 1985.

[22] "University of Oregon Route Views Project," `http://www.routeviews.org`.

[23] "BGP reports," `http://bgp.potaroo.net`.

[24] Tao Wan, Evangelos Kranakis, and P.C. van Oorschot, "Pretty secure BGP (psBGP)," in *Proceedings of Network and Distributed System Security Symposium (NDSS 2004)*, February 2004.

[25] Yih-Chun Hu, David McGrew, Adrian Perrig, Brian Weis, and Dan Wendlandt, "(R)Evolutionary bootstrapping of a global PKI for securing BGP," in *Proceedings of the 5th Workshop on Hot Topics in Networks (HotNets-V)*, Irvine, CA, USA, November 2006.

[26] R. White and B. Akyol, "Deployment considerations for Secure Origin BGP(soBGP)," *Internet Draft, Internet Engineering Task Force*, June 2003.

[27] Geoffrey 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 *Proceedings of Network and Distributed System Security Symposium (NDSS 2003)*, February 2003.

[28] Josh Karlin, Stephanie Forrest, and Jennifer Rexford, "Pretty Good BGP: Improving BGP by cautiously adopting routes," in *Proceedings of the 14th IEEE International Conference on Network Protocols (ICNP 2006)*, November 2006.

[29] E yong Kim, Li Xiao, and Klara Nahrstedt, "Identity-based registry for secure inter-domain routing," in *Proceedings of ACM Symposium on Information, Computer and Communications Security' 2006*, March 2006.