Symmetric Key Approaches to Securing BGP—A Little Bit Trust Is Enough

Bezawada Bruhadeshwar, *Member*, *IEEE*, Sandeep S. Kulkarni, *Member*, *IEEE*, and Alex X. Liu

Abstract—The Border Gateway Protocol (BGP) is the de facto interdomain routing protocol that connects autonomous systems (ASes). Despite its importance for the Internet infrastructure, BGP is vulnerable to a variety of attacks due to lack of security mechanisms in place. Many BGP security mechanisms have been proposed. However, none of them has been deployed because of either high cost or high complexity. The right trade-off between efficiency and security has been ever challenging. In this paper, we attempt to trade-off between efficiency and security by giving a little dose of trust to BGP routers. We present a new flexible threat model that assumes for any path of length h, at least one BGP router is trustworthy, where h is a parameter that can be tuned according to security requirements. Based on this threat model, we present two new symmetric key approaches to securing BGP: the centralized key distribution approach and the distributed key distribution approach. Comparing our approaches to the previous SBGP scheme, our centralized approach has a 98 percent improvement in signature verification. Our distributed approach has equivalent signature generation cost as in SBGP and an improvement of 98 percent in]signature verification. Comparing our approaches to the previous SPV scheme, our centralized approach has a 42 percent improvement in signature generation and a 96 percent improvement in signature verification. Our distributed approach has a 90 percent improvement on signature generation cost and a 95 percent improvement in signature verification cost. We also describe practical techniques for increasing the long-term security and collusion resistance of our key distribution protocols without increasing the signature generation and verification costs. By combining our approaches with previous public key approaches, it is possible to simultaneously provide an increased level of security and reduced computation cost.

Index Terms—Border gateway protocol, symmetric key distribution protocols, routing security, collusion resistance.

1 Introduction

The Internet consists of independently administered networks, which are called autonomous systems (ASes). The Border Gateway Protocol (BGP) is the de facto interdomain routing protocol that connects ASes together [2]. BGP provides two essential services: mapping IP prefixes onto the ASes that own them and the construction of source specific paths to each reachable prefix. Every BGP router announces the IP prefixes that its AS owns in an update message and sends the message to its neighboring BGP routers. Received update messages are recursively concatenated with an additional AS number and propagated from AS to AS forming a routing path, which will be used to forward traffic. When a BGP router receives multiple paths for the same prefix, the router chooses the best path based on multiple criteria such as path length, routing policies, etc. Although one AS may have multiple BGP routers, all BGP routers within the same AS use the same AS number. For simplicity, in this paper, we use the three terms "AS," "BGP router," and "router" interchangeably when there is no confusion.

- *B. Bruhadeshwar is with the International Institute of Information Technology, Hyderabad 500032, India. Email: bezawada@mail.iiit.ac.in.*
- *S.S. Kulkarni and A.X. Liu are with the Department of Computer Science and Engineering, Michigan State University, East Lansing, MI 48824. Email: {sandeep, alexliu}@cse.msu.edu.*

Manuscript received 5 Apr. 2009; revised 25 Feb. 2010; accepted 5 July 2010; published online 3 Jan. 2011.
Recommended for acceptance by C. Hu.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2009-04-0153.
Digital Object Identifier no. 10.1109/TPDS.2011.19.

The BGP update messages are undoubtedly important as they enable ASes to construct a consistent view of the network topology. Invalid update messages may result in incorrect routing tables, which could lead to three types of potentially disastrous consequences. First, incorrect BGP routing tables may make a range of IP addresses unreachable, which constitutes a denial-of-service attack. Second, incorrect BGP routing tables may make some packets travel through a malicious BGP router, which may launch man-in-the-middle attacks by eavesdropping, tampering, inserting, or dropping messages. Third, incorrect BGP routing tables may make some packets travel more hops than necessary to reach their destination, which degrades the Internet routing performance. However, due to the lack of security mechanisms in the current BGP protocol, attackers may spoof or tamper BGP messages. Thus, it is critical for a recipient AS to validate the authenticity (i.e., to detect spoofing) and integrity (i.e., to detect message tampering) of update messages before making routing decisions. For simplicity, in the rest of the paper, we use "BGP updates" to mean "BGP update messages." We refer the process of validating the authenticity and integrity of BGP update messages as "BGP path validation."

Many solutions for securing BGP have been proposed previously (e.g., [3], [4], [5], [6], [7], [8], [9], [10]). However, none of them have been adopted so far due to either high cost (such as S-BGP that extensively uses public key cryptography operations [5]) or high complexity (such as SPV that requires complex state maintenance and fairly large computational resources for BGP routers where such resources are of critical value [10]). In examining previous

solutions, we observe that for any given advertised path, these solutions require all nodes on that path to validate the prefix of the path up to that node. This constitutes the root cause of the inefficiency and complexity of previous solutions. Actually, this may be unwarranted for every path advertisement because BGP routers are expected to be more trustworthy than end hosts. BGP routers are typically owned by large Internet service providers (ISPs) that have little incentive in intentional falsification of route advertisements. Although compromising a BGP router may be possible, compromising multiple BGP routers on the same path at the same time by the same attacker is unlikely. Actually, it has been observed that the majority of incorrect BGP update messages are not caused by attackers, instead, they are caused by the misconfiguration of BGP operators [11]. Furthermore, it has been observed that most of the paths on the Internet remain stable [12] due to policy decisions or route stability, that is, the update messages follow more or less the same paths.[1] Thus, if we use a trust building approach along a particular path, then, even a few trusted nodes can eliminate the effect of the malicious routers as we know that the update message would be processed by trusted routers at some point. However, implicitly trusting all routers is also unreasonable since some routers could be compromised.

The above observations suggest a threat model where the number of malicious routers on any given path is limited. In particular, we consider the model where for any path of length h, at least one BGP router is trustworthy. Of course, we do not know which router is trustworthy. For ease of presentation, consider the case where we have a trustworthy BGP router X. Then, before advertising any path that includes X, X would have checked the authenticity and integrity of the path up to X. If X is trustworthy, then each subsequent node on the path, which receives the path advertisement containing X, does not need to validate the path before X, instead, it only needs to validate the path from X up to itself. In other words, instead of including a verification from each node on the advertised path, it would suffice if only signatures from X onward are used. Of course, this new scheme must accommodate that we do not know which routers are the actual trustworthy ones.

So far, we have discussed the idea of reducing the cost of validating BGP paths by reducing the number of validation operations needed for each path. Another cost in BGP security is the cost of verification itself. Existing approaches in [5], [6], [7] use digital signatures constructed using public key cryptography. However, such digital signatures are expensive to generate and verify, which consequently degrade the performance of BGP routers. Furthermore, digital signatures require public key and certificate management, which requires additional infrastructure from the network and adds to the processing overhead of BGP routers [13], [14]. The performance of BGP routers is a critical concern as the volume of traffic passing through BGP routers is very high [15]. Due to this fact, a BGP router needs to process update messages in the shortest time possible to avoid route disruptions and dropping/delaying of packets. Hence, there is a need for lightweight symmetric key based mechanisms that retain the benefits of digital signatures, authentication, integrity, and nonrepudiation, while maintaining the good performance of BGP routers.

Based on our above two ideas, one reducing the number of path validation costs by adding a little bit of trust into our threat model and one reducing the cost of each path validation by using efficient symmetric key management schemes, we propose two new symmetric key approaches to securing BGP. The first approach is a family of centralized key management protocols for securing BGP, which require a trusted server for distributing keys. Each BGP router only needs to maintain $O(\log^2 N)$ keys where N is the total number of BGP routers. The verification cost is even lower, $O(\log N)$ or less. The second approach is a family of distributed key management protocols for securing BGP, which does not require a trusted server for distributing keys. Distributed key management protocols are initiated by individual senders who intend to provide authentication for their messages. The storage cost in these schemes is higher than centralized key management protocols. The signature and verification costs in these protocols are also $O(\log N)$. The small signature and verification cost makes these distributed key management protocols attractive if performance is the primary concern for BGP routers. Another advantage of the distributed key management protocols is that they do not require the deployment of a trusted server, which makes deployment easy. We evaluate the performance of our protocols against standard public key cryptographic solutions as well as symmetric key solutions that have been proposed for securing BGP. We show that our solutions perform orders of magnitude better than public key solutions.

One important concern with deploying symmetric key distribution protocols is the long-term security of the symmetric keys and the effect of colluding users. Long-term security of symmetric keys is important as the keys are prone to brute-force attacks and cannot be used for long periods. To address long-term security, we describe an epoch-based key distribution where each epoch represents a time frame. The BGP speakers use different signature keys during the different epochs. The epoch-based key distribution reduces the lifetime of the symmetric keys and thereby, increases the strength of the protocol against active attacks. The issue of colluding users has been the attention of most symmetric key distribution protocols where a group of malicious users pool their keys to compromise some or all of the system keys. For example, the colluding users can use their collective keys to forge the signatures of a legitimate sender. To address this issue, in our key distribution protocols, we generate a pseudorandom permutation of the users for each epoch. The keys given to a user depend on its position in the permutation. This approach has the effect of diluting collusion. For example, if a set of users are successful during a particular epoch, there is no guarantee that the same set of users can succeed during a different epoch. This causes the users to increase the size of the colluding set which, is a nontrivial task. We demonstrate the effectiveness of our approach through experimental analysis.

1. In [12], based on BGP data from 40 global ASes, Butler et al. observed that on average 67-98 percent of paths were stable over the period of one year and over 99 percent paths were stable over a period of one month.

The rest of this paper proceeds as follows: In Section 2, we review BGP and present our threat model. In Section 3, we present the technical details of our centralized key management protocols and distributed key management protocols. In Section 3.3, we compare the relative merits and demerits of the various key management protocols. In Section 4, we describe techniques to improve the strength and the collusion resistance of our protocols. Our experimental results are shown in Section 5. We analyze the security of our proposed approaches in Section 6 and discuss deployment concerns in Section 7. We conclude in Section 8.

2 Background and Related Work

In this section, we give a brief overview of the BGP protocol [2], outline the security issues in current BGP implementations, discuss previous work, and describe our threat model and assumptions.

2.1 BGP Overview

The main objective of BGP is to advertise the routing path information for IP prefixes. We assume that each AS has a designated router (speaker) which communicates with BGP routers from neighboring ASes. Toward this, BGP routers initiate TCP connections with other BGP peers and exchange the path information in the form of BGP update messages. We note that, an AS might contain other EBGP routers and IBGP routers as well. In this work, unless stated otherwise, the term "router" and "BGP router" refers to the designated BGP speaker(s) for that AS. Thus, communication between two ASes implies that a BGP router from one AS is communicating with a BGP router from the other AS.

For this discussion, we represent an update message as a tuple: (*prefix*, *as_path*), where the *prefix* denotes what the message needs to advertise or withdraw, and the *as_path* denotes the sequence of ASes through which this update message has traversed. When a BGP router receives an update message, it will concatenate the *as_path* field of the message with its AS number and propagate the message to other neighboring ASes. The information in the *as_path* field is critical for detecting routing loops and deciding the best forwarding path for IP prefixes. Although BGP update messages can be used to advertise as well as withdraw IP prefixes, without loss of generality, we assume that update messages contain prefix advertisement. All our discussion applies to withdraw messages as well.

We illustrate the important functionalities of update messages using seven ASes that are interconnected as shown in Fig. 1. In this example, we use letters A through G to denote the seven ASes. Fig. 1 shows the traversal of the BGP update message originated from AS A, who owns the IP prefix 24.0.0.0/8, denoted as P. To advertise that A owns prefix P, A sends (*P, A*) to its neighboring ASes, including B. When B receives this message, it first updates its routing tables appropriately, then prepends its AS number to the *as_path* field of the tuple and sends the new update message (*P, BA*) to its neighboring ASes including C and D. Proceeding in this manner, the update message is propagated until all the ASes are aware of the path to reach the prefix advertised by *A*. Note that when G receives two update messages (*P, ECBA*) (*P, FDBA*) for the same prefix P, G in this example chooses (*P, FDBA*) based on its local routing policy and the business relationships with other ASes.[2] Now, if any user in *G* has data to send to any computer whose IP address falls in the advertised prefix P, *G* will forward the data to *F* which will then forward to D, then D forwards the data to B, and B finally forwards the data to A.

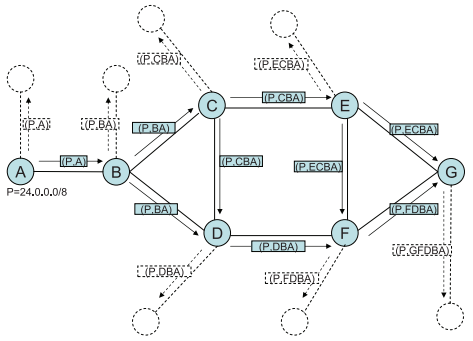


Fig. 1. Traversal of BGP update messages.

2.2 BGP Security Issues

There are four major types of attacks on BGP control messages: *deletion, replay, modification, and insertion*. The first two types of attacks are out of the scope of this paper. Deleting BGP control messages seems indistinguishable from legitimate route filtering [7]. Replay can be handled by setting expiration time for BGP messages [7]. This paper concerns the latter two types of attacks: modification and insertion. In path modification attacks, adversary changes the routing information of the BGP update message with a malicious intent. In BGP path insertion attacks, also called path forgery attacks, an adversary forges a path to divert traffic from legitimate destinations. We refer to both BGP path modification and forgery attacks as BGP path falsification attacks.

There are four types of BGP control messages: *open, keepalive, notification*, and *update*. The first three are used by BGP to establish and maintain BGP sessions with their peers. As stated by Hu et al., these three types of messages can be protected by a point-to-point authentication protocol such as IPSec [16]. Since these messages are meaningful only between the two communicating BGP speakers and the content of these messages is constant throughout the current session any point-to-point authentication protocol can provide security.

This paper concerns protecting the BGP update messages. Each update message can be viewed as a broadcast message that is sent out to every possible BGP speaker. Since IPSec can only protect a session between two nodes, it is unsuitable to protect the update message which is sent by one sender to many possible recipients. One additional aspect that is important is that each BGP node modifies the *as_path* information in the update message. Hence, there needs to be a mechanism that enables us to verify the information added by each BGP speaker. In particular, we are concerned with the authenticity (to protect message insertions) and

2. Unlike other routing protocols, BGP does not primarily use shortest path as a parameter for selecting the best route.

TABLE 1
Comparison for Various BGP Security Solutions

Feature	SBGP [5]	SPV [10]	soBGP [6]	psBGP [7]	IRV [4]	Whisper [3]	Ours
Path Integrity	Yes	Yes	No	Yes	Yes	Partial	Yes
Origin Authentication	Yes	No	Yes	Partial	Yes	No	No
Forgery/Replay Detection	Yes	Yes	Partial	Yes	Yes	Yes	Yes
Collusion Resistance	Yes	Partial	Partial	Partial	Yes	Yes	Partial
DOS Attack Resilience	Partial	Partial	Yes	Partial	Partial	Yes	Partial
Update Size Change (current Max. 4KB)	Increases	Increases	None	Increases	None	None	Increases

integrity (to protect message modification) of BGP update messages. In BGP path modification attacks, an adversary may add, remove, or alter AS numbers from the *as_path* field of BGP update messages. The goal of BGP path modification and forgery attacks is to influence packet routing in a way that benefits the attacker. The important falsification attempts are 1) either to remove some existing ASNs from the update and 2) to add some new ASNs into the update message.

In this work, we only focus on the protecting the content of the message regardless of the intent of the nodes in propagating such messages. For example, if a falsified update is propagated by a misconfigured BGP node, we treat the message as a possible subversion and deal with it accordingly. Determining the actual intent and source of falsification is nontrivial due to threats such as IP spoofing.

2.3 Past Solutions for BGP Security

BGP security has been a focus of studies for sometime.[3] In [8], the authors describe a solution in which the BGP data exchanged by two BGP peers is encrypted by a session key. Also, the authors use sequence numbers and time stamps to achieve total ordering of messages which protects against session hijacking and replay attacks. However, this scheme involves expensive digital signature verification by each intermediate router since part of the message is signed by the originating router.

In [5], Kent et al. present S-BGP, a comprehensive framework for achieving security in BGP using two Public-key Infrastructures (PKIs). To validate an update, the originator of the update message, signs the IP prefixes using its private-key and sends the update to its neighboring routers. Upon validating the message through signature verification, the neighboring router creates a *route attestation*, i.e., it updates the *as_path* field, signs it with its private key and appends it to the original message to create the new update. Every transit router verifies all the attached signatures and adds its own route attestation to the update message. Thus, the update is protected against path modification and forgery attacks in this manner. However, S-BGP places significant computation overhead on the BGP routers since digital signature creation and verification are costly as studied in [14] and degrade performance of the BGP routers.

In secure origin BGP (soBGP) [6], White describes a PKI-based solution that requires three public-key certificates, two of which are similar to those used in S-BGP. The soBGP scheme reduces the cost of signature verification by verifying long standing information such as address ownership, organizational relationships and topology, before the BGP sessions start, and storing this information at the routers. However, as in S-BGP, soBGP also incurs significant signature verification cost and an additional cost of storing the topology information. In [7], Van Oorschot et al. describe the *Pretty Secure BGP* (psBGP) which uses a centralized trust model for AS number authentication and decentralized trust model for verifying IP address ownership. However, the semantics of psBGP are still based on PKI and hence, are computationally expensive.

In [9], [10], Hu et al. describe a scheme based on Merkle-hash trees [18], [19] and one-way hash chains, to preserve path vector integrity for routing protocols. In [10], the authors use one-time signatures for protecting the *as_path* field. However, although SPV is efficient compared to public-key based solutions, it involves significant precomputation and state overhead. For example, each router needs to keep track of the one-way hash chain values and generate the values that need to be revealed to the next hop router.

In [3], the authors describe the *Listen* and *Whisper* protocols which describe mechanisms to detect routing anomalies. These protocols, however, do not provide explicit route authentication which is necessary in the current Internet scenario. In [4], the authors propose a security architecture called *Interdomain Routing Validator* (IRV). In IRV, each AS designates a data server that can answer queries about the routing information. However, IRV requires an additional secure mechanism, like IPSec, to secure the query-answer sessions and in addition requires the data to be updated in a timely manner at the IRV server. Note that the scope of this paper is on BGP control plane security, not BGP data plane security [20]. Recently, Hu and Mao have proposed methods to use data plane information to validate occurrences of IP hijacking in real time [21].

Feature evaluation of our protocols. In our work, we describe two key distribution protocols, *centralized*—where a central authority is responsible for key distribution and *distributed*—where the individual router is responsible for key distribution, to secure the BGP updates. Our protocols secure the integrity of UPDATE messages by using symmetric key-based signatures. A brief feature based evaluation of our protocols is given in Table 1.[4]

2.4 Threat Model and Assumptions

Our threat model is based on the various falsification attacks [17] on the BGP protocol, some of which have been detailed

3. For a comprehensive survey of proposed BGP security measures, the reader can refer to [17].

4. The *Partial* attribute denotes that under certain conditions the security solution is weakly secure.

in Section 2.2. From these attacks, the types of falsification attacks that we address in this work are: generation of false update messages by spoofing source IP address, insertion, or deletion of AS numbers from the *as_path* field, and changing the order of AS numbers in the *as_path* field. Note that, a combination of these attacks is also possible. We address such combined attacks as well.

We assume that one or more BGP routers could be malicious. However, we assume that there is at least one nonmalicious node along a given path of length h. We treat any misconfiguration of BGP routers as malicious and accordingly address this from the point of view of falsification.

Furthermore, we assume the existence of a public-key infrastructure (PKI), similar to that described in S-BGP [5], that enables two essential functionalities. First, the PKI infrastructure allows a centralized authority like ICANN to authorize the announcement of a prefix by providing the necessary public-key certificate to the current owner of the prefix. The BGP speaker which announces the prefix, signs the prefix with its private-key, includes the certificate with the signature and sends the update message. Any BGP receiver can verify the signature by extracting the public-key of the origin BGP speaker from the included certificate. Note that, each receiver only needs to verify one public-key signature which is also required in other techniques like SBGP, soBGP, and psBGP. Second, the PKI infrastructure can be used to bootstrap the BGP speakers with the necessary symmetric keys as required in our key distribution protocols. The public-keys of the BGP speakers can be used to secure the distribution of the keys using the techniques described in Section 7.1.1.

3 Symmetric Key Management for Securing BGP

We examine two types of symmetric key distribution approaches for securing BGP messages. In the first approach, a centralized controller establishes the necessary keys among the BGP routers and hence, we call protocols using this approach as *centralized key distribution* protocols. In the second approach, we assume that a centralized controller does not exist and each AS distributes the necessary keys to the BGP routers of other ASes. We call key distribution protocols using this approach as *distributed key distribution protocols*.

3.1 Centralized Key Distribution Protocols

We describe the square grid key distribution protocol [22] in Section 3.1.1. Then, in Section 3.1.2, we describe how the square grid protocol can be used to achieve security of the BGP update messages. In Section 3.1.3, we describe a trust model that reduces the signature block size in each update message. Finally, in Section 3.1.4, we describe extensions of grid protocol that provides similar properties while reducing the number of keys.

3.1.1 The Square Grid Protocol

In the square grid protocol [22], n users are arranged in a *logical* square grid of size $\sqrt{n} \times \sqrt{n}$. Note that in the grid protocol (and other protocols we describe) the topology is

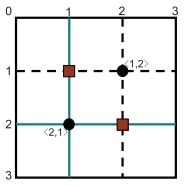


Fig. 2. Example square grid.

logical. Our protocols are not topology dependent and assume that the Internet is modeled as a connected graph, i.e., each node is acting as a sender and all the remaining nodes are the receivers. Two nodes need not be directly connected to each other and can talk via other nodes. There is no restriction on which BGP router can act as a sender or a receiver.

Now, each location, $\langle i, j \rangle$, $0 \leq i, j < \sqrt{n}$, in the grid is associated with a *user* $u_{\langle i,j \rangle}$ and a *grid key* $k_{\langle i,j \rangle}$. Each user knows all the grid keys that are along its row and column. For example, in Fig. 2, the grid key associated with $\langle 1, 1 \rangle$ is known to users at locations $\langle j, 1 \rangle, \langle 1, j \rangle, 0 \leq j \leq 3$. Additionally, each user maintains a direct key with users in its row and column. This direct key is not known to any other user.

Now, consider the case where user A wants to set up a session key with user B. Let the locations of A and B be $\langle j_1, k_1 \rangle$ and $\langle j_2, k_2 \rangle$, respectively. In this case, A selects the session key and encrypts it as follows: if A and B are in same row or column, A uses the unique key between A and B for session encryption. Otherwise, A uses an XOR of grid keys at locations $\langle j_1, k_2 \rangle$ and $\langle j_2, k_1 \rangle$.

For example, in Fig. 2, the users marked at location $\langle 1, 2 \rangle$ and $\langle 2, 1 \rangle$ use the keys marked with ■. Along with the encrypted session key, A also sends its own grid location (in plain text) to B. The above key selection protocol ensures that, in the absence of collusion, the key used by A cannot be derived by any other user other than A and B. Hence, the above protocol can be used for establishing the session key. (cf. [22] for proof.)

3.1.2 Square Grid for Authentication in BGP

The goal of the authentication in BGP is to ensure that an AS receiving the route update can authenticate the source of the advertisement and the integrity of the *as_path*.

First, we focus on the problem of authenticating the source of the advertisement. Note that the square grid protocol is designed for secure and authenticated point-to-point communication between two nodes. However, in BGP, the same path advertisement may be sent to several neighbors. Moreover, this path advertisement may be further forwarded by their neighbors, and so on. Due to these reasons, for this discussion, we assume that a given path advertisement may be possibly verified by any AS in the network.

To use the grid protocol, each AS is assigned a logical identifier in the grid and the corresponding keys as specified by the grid protocol. We assume that a centralized controller has assigned the logical identifiers and the corresponding keys to the BGP routers. Note that, the process of key establishment can be achieved by the use of public-keys or other similar key agreement protocols.

Now, consider the case where an AS with logical identifier $\langle j_1, k_1 \rangle$ needs to advertise a route $\langle A_1 A_2 \dots A_n \rangle$ for prefix $24.12.0.0/8$. Toward this end, $\langle j_1, k_1 \rangle$ hashes the message $\langle \langle A_1 A_2 \dots A_n \rangle, 24.12.0.0/8 \rangle$ using each of the keys it has (both direct and grid keys) separately. Subsequently, to advertise the route, it sends a packet consisting of the following information: 1) its ID, namely, $\langle j_1, k_1 \rangle$, in plaintext, 2) the route $\langle A_1 A_2 \dots A_n \rangle$ and prefix $24.12.0.0/8$ in plaintext, and 3) (hash value of) the message obtained by hashing $\langle \langle A_1 A_2 \dots A_n \rangle, 24.12.0.0/8 \rangle$ with each of the keys it has. This packet is denoted as the *signature block* of the message.

Whenever an AS receives this message, it uses the ID, say $\langle j_1, k_1 \rangle$, associated with the message to determine which keys should be used for authentication. In particular, as specified in Section 3.1.1, it identifies a collection of grid keys or a direct key that it would use if it were to communicate with an AS with logical identifier $\langle j_1, k_1 \rangle$. Then, using those keys separately, it hashes $\langle \langle A_1 A_2 \dots A_n \rangle, 24.12.0.0/8 \rangle$ (received in plaintext). It determines if all the hash values it computed are present in the signature block. If so, it accepts the message.

Theorem 1. *The above approach ensures that when an AS accepts a message that contains ID $\langle j_1, k_1 \rangle$, the corresponding message is indeed sent by node $\langle j_1, k_1 \rangle$.*

Now, consider the network shown in Fig. 1. In this figure, node C advertises the route CBA. When node E receives this message, it needs to ensure that BA was sent by B and CBA was sent by C. This can be achieved by having node C concatenate the signature block of B and its own signature block for route CBA and send it to node E. Upon receiving this message, node E can verify the route advertised by B and C.

3.1.3 Reducing Signature Block Size Based on Trust between ASes

Similar to the algorithms that use public-keys (e.g., [5]), the size of the signature block increases with path length. In this context, we note that while perfect authentication is desirable for BGP routing messages, the ASes differ from individual users on the Internet. In particular, while an individual AS may be compromised, we do not anticipate a significant misrepresentation to be done by ASes. Hence, as discussed in Section 2.4, we consider the threat model where at least one AS on any path of length h is trustworthy (although the exact trustworthy AS is unknown).

For sake of presentation, we consider Fig. 1 and let $h = 2$. Consider the case where node E advertises the route $ECBA$ and this route is received by G. Based on our assumption, either AS C or E (or both) is trusted. Now, we show that in this case, node G does not need to receive the signature block of B. To see this, we consider two cases:

- E is trustworthy. In this case, AS E has verified the validity of route BA and CBA. AS G can verify that the route $ECBA$ is advertised by E. Since BA is a prefix of route $ECBA$, node G does not need to receive signature block of B.
- C is trustworthy. In this case, AS C has verified the validity of route BA. AS G can verify that CBA is indeed advertised by C using the signature block of C. Hence, it does not need the signature block of B.

We can generalize the above scenario for arbitrary paths and arbitrary values of h. In particular, if at least one of h consecutive ASes is trusted then the signature block of the last h ASes needs to be attached with the route. (If the path length is less than h then all signature blocks would need to be attached.)

Providing additional security. If an AS desires an additional level of security then it can request additional signature blocks as needed. To illustrate this, consider the network in Fig. 1. When node G receives the signature block of CBA and $ECBA$ and desires an additional level of security (than that provided in the case where at least one of two consecutive ASes can be trusted), it can do so by obtaining the signature block of AS B. Note that the signature block can be obtained by the last node on the path it receives, namely E, rather than obtaining it directly from the source, namely B. In general, if an AS desires additional level of security, it can obtain the corresponding signature blocks from the nodes on the advertised path it receives.

Optimization. From the study of Internet characteristics [12], it is known that as much as 67-98 percent of BGP paths remain stable for considerable length of time. Hence, it is reasonable to assume that a BGP router, which is sending route advertisement, is aware of some transit nodes downstream. This knowledge could be collected and cached over a time period by observing message exchanges, e.g., when BGP update messages are propagated in the forward direction (downstream) or when data are exchanged in the reverse direction (upstream). With this knowledge, the number of signatures that are used by the BGP router can be reduced. Since some knowledge of routers along a downstream path is available, the BGP router can include only those signatures that can be verified by these routers. Other signatures which are not useful for verification can be eliminated without loss of security. Note that, a similar approach was used by [10], to achieve per-hop verification of the BGP update messages.

3.1.4 Storage-Efficient Key Distribution Protocols

Recently, in [23], [24], [25], the authors describe storage efficient key distribution protocols for achieving confidentiality and authentication in completely connected communication networks. Essentially, these protocols maintain a higher dimension grid to reduce the number of keys used in them. Of these the protocol in [23] uses $4 \log^2 N$ keys, the protocol in [24] improves it to $\log^2 N$, and the protocol in [25] improves it to $\frac{1}{2} \log^2 N + O(\log N + \log \log N)$. All these protocols provide a property similar to that of the grid protocol. In particular, if a node sends a message that includes a signature from each of the keys it has and the receiver verifies the signatures based on the common keys then it can conclude that the message is authentic. Because these protocols use grids with dimension of $\log N$, whenever the node receives a message, it needs to verify two signatures from each grid. The most storage efficient protocol from these protocols is from [25]. One can choose the appropriate protocol depending on the type of storage and verification requirements. Later, we compare these protocols with

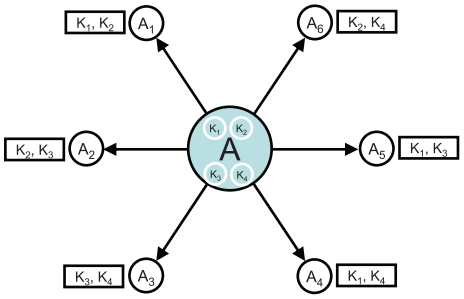


Fig. 3. Example key distribution for star network.

public-key cryptography solutions and show that, our approach in Section 3.1.2 reduces the authentication cost substantially.

3.2 Distributed Key Distribution Protocols

In distributed key distribution protocols, each node is responsible for locally generating and distributing the keys to the other users in the network. This approach is especially useful when establishing a centralized key distribution infrastructure is difficult. In [25], [26], the authors describe key distribution protocols for a star communication network. In star communication networks, a center node communicates with several satellite nodes and vice versa. The satellite nodes do not communicate with each other. In Section 3.2.1, we describe the key distribution protocol from [25] and show that this protocol provides message authentication.

3.2.1 Optimal Key Distribution for Star Networks

In the key distribution protocols described in [25], the center node maintains a set of k keys. Each satellite node receives a unique subset of size l from this set. Note that, by construction, no two satellite nodes have identical subsets of keys. We term this protocol instance as $p(k,l)$. For example, see Fig. 3 that shows the instance $p(4,2)$. Using $p(k,l)$, authentication of messages can be achieved in the communication as follows.

To authenticate a message m broadcasted by the center node to the satellite node, the center node generates and transmits authentication codes with each of the k keys. Now, when a satellite node receives this message, it uses its subset of l keys to compute l authentication codes. The satellite node then verifies these authentication codes with the corresponding authentication codes sent by the satellite node. Note that, each satellite node can verify only those authentication codes for which it has the corresponding generating key.

In [25], authors have shown that given a set of n satellite nodes, maintaining $k = \log n + 1/2 \log\log n + 1$ secrets at the center node is sufficient if each node receives $k/2$ keys. If each node receives $k/2$ keys then there exists a set of two nodes whose collusion can reveal all the keys. Hence, to deal with this case, we can assign each node only k/m keys where m is the level of desired collusion resistance. For example, if we choose $m = 10$ then maintaining 40 keys and letting each node receive 4 would allow $C(40,4) = 91{,}390$ satellite nodes. And, this would be sufficient for BGP which currently has approximately 30,000+ ASes [27].

3.2.2 Using $p(k,l)$ for Authentication in BGP

A BGP network with N routers can be viewed as a collection of N star networks where each BGP router is the center node for one of these networks. Now, each BGP router runs an instance of $p(k,l)$ as described in Section 3.2.1. To authenticate an update message, the originating BGP router generates the necessary authentication codes from its keys. The generation and verification approach of the signature block is similar to that for centralized protocols except that the keys used in signature generation and verification are dictated by the above algorithm.

3.3 Comparison of Key Distribution Protocols and Issues

We compare the performance of centralized key distribution protocols with distributed key distribution protocols. For discussion, we only consider the key distribution protocols from [25]. In distributed key distribution protocols, the number of keys maintained by each sender is high, $O(N \log N)$ keys for N nodes but the number of keys used to sign the messages is small $O(\log N)$ signatures. On the other hand, in centralized protocols, the storage at each node is much small $O(\log^2 N)$ keys but the signature cost is higher $O(\log^2 N)$ signatures. Note that, in both the protocols, the cost of signature verification is comparable, $O(\log N)$ signatures are verified. The main difference between these two protocols is in the terms of difficulty of deployment. In most practical scenarios, for centralized key distribution protocols, it is difficult to have a trusted central controller for key distribution and management. Moreover, a central controller is a single point of vulnerability and may not be desirable in many scenarios. Whereas in the distributed key distribution protocols, key distribution and management is relatively easy as each individual node maintains and distributes the necessary keys. However, in distributed key distribution protocols, a small percentage of colluding nodes can compromise several sender keys whereas this is not the case in centralized key distribution protocols which exhibit higher collusion resistance [25]. We note that, the distributed key distribution protocols lend themselves to incremental deployment as BGP nodes wishing to secure their communication may deploy these protocols without requiring support from external entities. Whereas, the centralized key distribution protocols may require support from the trusted entities on the Internet such as ICANN or IANA for large scale deployment. Note that, in spite of deployment difficulties, centralized key distribution protocols are desirable when storage needs to be reduced while maintaining good performance at the BGP routers.

One of the issues with shared symmetric key distribution protocols is the long-term security of the symmetric keys. The security of the symmetric keys is reduced if they are used over long time periods. Another issue with such protocols is collusion resistance. Since the key distribution protocols rely on shared symmetric keys, this issue needs attention. We address these issues in the next section and describe our solutions to these problems.

4 Extensions for Stronger Security and Collusion Resistance

In this section, we describe extensions to our approaches to improve the security and the collusion resistance of our

protocols[5] while keeping the signature and verification cost as before. In Section 4.1, we describe the key distribution strategy for maintaining the long-term security of the symmetric keys without increasing signature or verification cost. Our solution trades-off storage for the additional security provided. In Section 4.2, we describe our approach to improve collusion resistance and evaluate its effectiveness in Section 4.3. Our solution for collusion resistance does not assume that the actual colluding users are known in advance and improves the collusion resistance in an information theoretic sense by using a practical model of collusion.

4.1 Increasing Security of Signatures

One way to increase the strength of the signatures is to refresh the symmetric keys which are used for signature generation. However, refreshing the keys frequently is a difficult task, especially given the scale of BGP. To address this issue, we use two notions: key predistribution and periodic rekeying. In key predistribution, the users are given a substantial number of keys to avoid frequent key updates. In periodic rekeying, the keys are changed at the beginning of each period which is sufficiently long. Briefly, our approach is as follows: initially, a sufficiently long key update period is chosen and sufficient keys are distributed at the beginning of the period. Next, each key update period is split into fixed-time epochs. In each epoch, the BGP updates are signed and verified with mutually exclusive sets of keys chosen from the keys distributed to the users. Since each epoch lasts only one hour, due to the strength of 128-bit keys, it will be difficult for attackers to subvert the signatures in the short time that the keys are valid.

4.1.1 Key Distribution in an Epoch

For this discussion, we assume a period of 24 hours which is split into 1-hour epochs. In each 24 hour period, the sender generates $48 \log N$ symmetric keys. These keys are split into mutually exclusive subsets of $2 \log N$ keys each. Each subset corresponds to the signature keys for a particular epoch. Using the distributed key distribution protocol, the sender distributes the keys to the receivers for all the epochs at the beginning of the 24-hour period. For example, in each epoch, the sender can initiate the $p(k, l)$ protocol by choosing $k = 2 \log N$ and $l = \log N$. Each user receives the corresponding keys in a separate secure channel.[6]

4.1.2 Signature and Verification Costs

It can be observed that the epoch-based key distribution only increases the storage at the users and not the signature and verification costs. Within a particular epoch only $2 \log N$ keys are used to sign an update message and $\log N$ keys are used to verify the message. This implies that the performance of the BGP speakers does not get affected by the epoch-based key distribution.

5. For this discussion we consider distributed key distribution protocols. The same analysis applies for centralized key distribution protocols as well with some trivial changes.

6. The actual distribution process may be achieved by encrypting the keys and storing them at a repository controlled by a central authority like ICANN or at some public servers. The keys can be retrieved by individual receivers without contacting the actual sender.

4.1.3 Storage Trade-Off

We note that, using the epoch-based key distribution, the storage at the BGP nodes is still manageable considering that symmetric keys are only 128-bits long. Furthermore, our signature generation and verification approach does not require all the keys to be used at the same time and hence, there is no excessive I/O activity that can degrade performance. For example, if the number N of BGP nodes is over $30{,}000 \approx 2^{15}$ nodes, then the storage at each node is: $(N \log N + 48 \log N) * 128$ bits, which is only about 7.5 MB of key storage. Each BGP node only needs to access this storage when the current epoch changes and hence, this storage does not affect the performance of the BGP node.

4.2 Increasing Collusion Resistance

First, we describe a model of collusion that is close to practice and then describe an approach that makes collusion in such a model difficult to sustain over long periods of time.

4.2.1 Model of Collusion

In shared key distribution protocols, users who collude usually do so only if the combined set of keys has very few common keys, i.e., if they collectively have the maximum possible information. Collusion in this manner ensures that the users have access to more keys and the number of users required for the collusion is the least possible. Furthermore, for such a colluding set, with high probability the colluding set does not change as time goes by as the users in the colluding set have no need to add new users without any clear benefit. Moreover, adding new users into the colluding set is a nontrivial task as there is the risk of being detected by legitimate nodes and thereby, exposing the original colluding set of users.

From this model, we can now identify some properties that the key distribution protocol needs to have to improve collusion resistance. The first property is that the key distribution protocol should require a relatively high number of colluding users to impersonate the sender. The second property is that, with the epoch-based key distribution, a set of colluding users should be ineffective over a majority of the other epochs. This means that, even if a set of colluding users are successful during a particular epoch, the same set of users should not be effective during other epochs. This implies that the keys assigned to the users need to be different across different epochs.

4.2.2 Increasing the Colluding Set

We note that in the distributed key distribution protocol, the minimum number of colluding users can be determined by the size of the subset of keys that is given to each user. For example in $p(k, l)$, the size of the colluding users is given by $\frac{k}{l}$. Note that this parameter can be increased by increasing the value of k and reducing the value of l. Intuitively, when the pool of keys is larger then the subset sizes required will also be lesser. Assuming that collusion can happen along any given path, a BGP speaker can choose these values accordingly. For example, in a $C(40, 4)$ distribution, the minimum colluding set is 10 users which is quite a large number in the given context.

4.2.3 Proposed Key Assignment

We describe a possible key assignment that satisfies the properties identified in the above discussion. Our approach

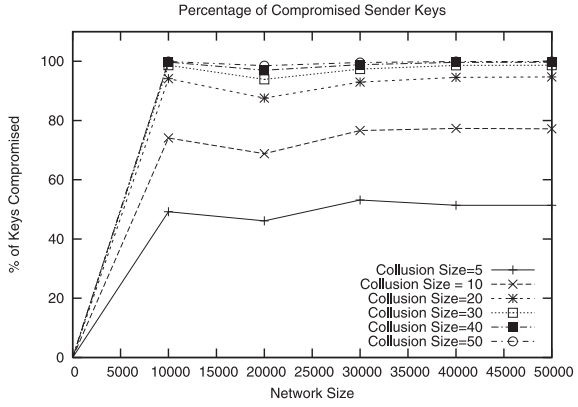


Fig. 4. Number of sender keys compromised over all epochs.

uses the concept of latin-squares to identify the subset of keys given to a particular user. A latin-square is a $N \times N$ matrix-like arrangement with a special property that along any row or any column each element occurs only once and each row/column is a permutation of the sequence $1, 2, 3 \ldots, N$.

Now, we assume that the users are arranged in a latin-square of size $N \times N$.[7] Each column in the latin square corresponds to the key distribution in a single epoch, i.e., the location of each user in each column identifies the keys that this user receives in that particular epoch. Since we have chosen 24 hours as the period, we select 24 rows from the latin-square and initiate the key assignment. Since each user occurs in a particular location only once along a column, for every epoch, each user receives a different subset of keys from the pool of keys being used. Since each user gets a different subset of keys in every epoch, a set of successful colluding users in particular epoch have to rely on chance that their collusion can break the sender secrets in the next epoch as well. If they find that their collusion is no longer effective then it implies that the users need to look for additional colluding users which is a nontrivial and risky task. Next, we experimentally show that, for practical purposes, this key assignment provides good collusion resistance.

4.3 Evaluating Collusion Resistance in Epoch-Based Key Distribution

We show the effectiveness of our scheme by considering the following parameters and evaluating the collusion among the same set of users across multiple epochs. We assume that within each epoch the sender uses a different pool of $2.\log N$ keys. To instantiate $p(k, l)$, we choose the smallest l that satisfies the $p(k, l)$ for this set of users. For example, for $N = 10{,}000$, the value of $l = 4$. The reason for this choice is that, the small number of keys per user requires a larger colluding set to break the sender keys. We have tried the effect of collusion on the epoch-based key distribution on group sizes 10,000 to 50,000. We have chosen the colluding set to be 5, 10, 20, 30, 40, and 50 users. In Fig. 4, we show the results of our experiments. We have found that for reasonable values of colluding users, i.e., ≤ 30 the collusion remains ineffective.

7. Techniques to generate latin-squares can be found in text books on combinatorics.

5 Evaluation

In this section, we evaluate the performance of our approach and compare it with S-BGP [5] and SPV [10]. Toward this, we evaluate the protocols on, the signature cost, verification cost, and storage overhead. In the centralized key distribution approach, we compare the protocols from [22], [23], [24], [25], with SBGP. Similarly, in the distributed key distribution approach, we compare the protocols from [25], [26] with SBGP. Next, we apply our trust model to centralized and distributed protocols and compare the results with SPV [10] and SBGP [5]. Finally, we show the performance of our protocols for different values of h.

We use the following notation to refer to the various key distribution protocols. We refer to the centralized key distribution protocols as follows: [22] as *Square Grid*, [23] as *Multiple Key Grids*, [24] as *Hierarchical*, [25] as *Star Hierarchical*. We refer to the distributed secret distribution protocols as follows: [26] as *Star Plain* and, [25] as *Star Optimal*.

Experimental methodology. In our simulation experiments, we focused on computing the number of operations required in each of these protocols. The number of operations provides an accurate means of comparison while ignoring implementation level variations. For the various symmetric key distribution protocols in our approach, we assume that a BGP router uses the HMAC (Hashed Message Authentication Code) algorithm with 512-bit keys to generate the corresponding 160-bit message authentication codes. The following simulation methodology was used.

- We used the path length as a parameter for experimentation. Typical path length varies from 5 to 16 hops.
- For SBGP, we assume that BGP routers use the RSA algorithm for signature generation and verification. We assume that the cost of signature verification using RSA is around 401 microseconds and that of a message authentication code is around 2 microseconds [10], in all our analysis.
- For SPV, we chose the number of leaves of the ASPATH protector, $\mathrm{n} = 256$ and the HORS signature parameter, $\mathrm{m} = 6$. These values give the highest security for a path length of 15 hops when using SPV.
- For each update, we counted the number of operations performed by each router using SBGP and SPV.

Note that, wherever necessary, we discounted some additional operations that may be required in these approaches, e.g., double hashing required in the advanced ASPATH protector used by SPV and so on. We did not consider the certificate verification cost and assumed that all such verifications are cached.

5.1 Comparison of Centralized Key Distribution Protocols with SBGP

In Figs. 5a and 5d, we compare the signature and verification cost of the different centralized key distribution protocols with SBGP. In SBGP, the cost of signature generation for a BGP speaker is only one signature, i.e., the route attestation that is added by this speaker. From Fig. 5a, we can observe that on an average, the cost of signature generation is lower

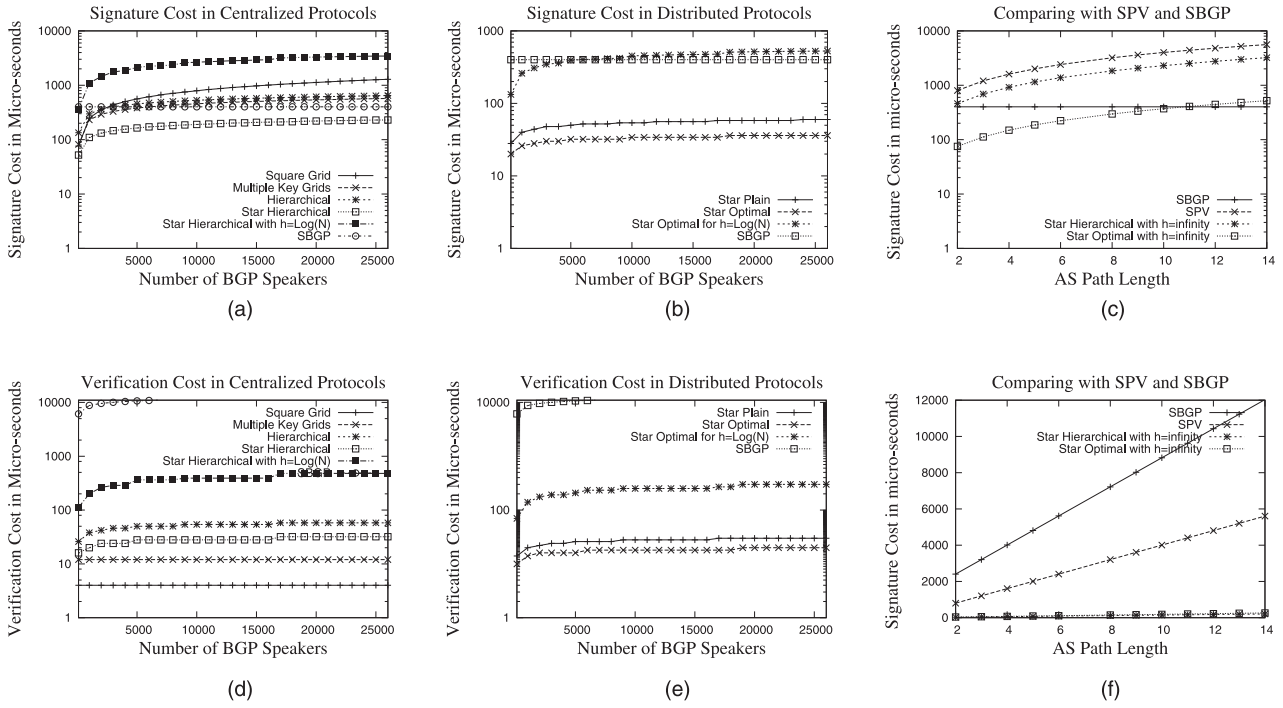


Fig. 5. Experimental results. (a) Signature cost in centralized key distribution protocols versus SBGP. (b) Signature cost in distributed secret distribution protocols versus SBGP. (c) Signature cost of star protocols with SBGP and SPV. (d) Verification cost in centralized key distribution protocols versus SBGP. (e) Verification cost of distributed secret distribution protocols versus SBGP. (f) Verification cost of star protocols with SBGP and SPV.

for the centralized protocols. The *Star Hierarchical* protocol has the lowest signature cost among all the protocols. We also show the signature cost incurred by using our trust model with the value of $h = \log N$, where N is the total number of BGP speakers in the system. We chose this value specifically, since most practical networks have logarithmic diameter and this value represents the maximum distance that an update can traverse. In other words, this is the cost incurred by the *Star Hierarchical* protocol when no intermediate node is trusted in the network.

In SBGP, the cost of signature generation is low as each BGP speaker only needs to add its own signature to the update. However, the cost of verification is high in SBGP, since each BGP will have to verify the route attestations of all the ASes that are part of the update message. For purposes of calculation of verification cost for SBGP, we assumed that each update traverses at the most $\log N$ ASes and computed the cost incurred by the last BGP router in the path.

From the results in Fig. 5d, we note that, the distributed protocols are orders of magnitude better than SBGP. Even for the case when $h = \log N$, in *Star Hierarchical* protocol, the cost of verification is lower than SBGP. In practice, for BGP routers, a smaller cost of verification is desirable. This is because the BGP router needs to make a decision whether or not to accept the update message and this can only be done after the verification process. Hence, improvement in the cost of verification is very important.

5.2 Comparison of Distributed Key Distribution Protocols with SBGP

Similarly, in Figs. 5b and 5e, we compare the signature and verification cost of the distributed secret distribution protocols with SBGP. From Fig. 5b, we observe that the signature cost of distributed protocols is much lower than that of SBGP. Furthermore, even when $h = \log N$, the signature cost is comparable with that of SBGP. As in the centralized protocols, the cost of verification in distributed protocols is orders of magnitude better than SBGP. We note that, the size of signature block in the distributed protocols is much smaller than in centralized protocols. This feature is very useful as the message authentication codes can be carried in the updates themselves without requiring additional message types to be introduced into the BGP protocol. These features indicate that distributed protocols lend themselves to incremental and scalable deployment.

5.3 Comparison of Signature and Verification Cost in Our Approach with SPV and SBGP

In SPV [10], each originating node needs to generate a one-time signature that will be verified by its downstream routers. The one-time signature is the root of a merkle hash-tree [18], [19], generated using the *as_path* field and a secret key held by the sender. This structure is called an *as_path* protector and is built over the number of ASes that need to be protected. For example, if security is required over 15 ASes then the *as_path* protector contains the one-time signatures of the 15 ASes. Hence, in SPV, each node needs to generate and verify several hash values. Thus, for SPV, the cost of signature generation and verification are comparable although verification is slightly smaller than signature generation. For comparing SPV, we chose the number of leaves of a single one-time signature to be 80 and the total ASes in the path to be 15 (cf. [10]). For clarity of presentation compare SPV and SBGP with the *Star Hierarchical*, which is a centralized protocol, and the *Star*

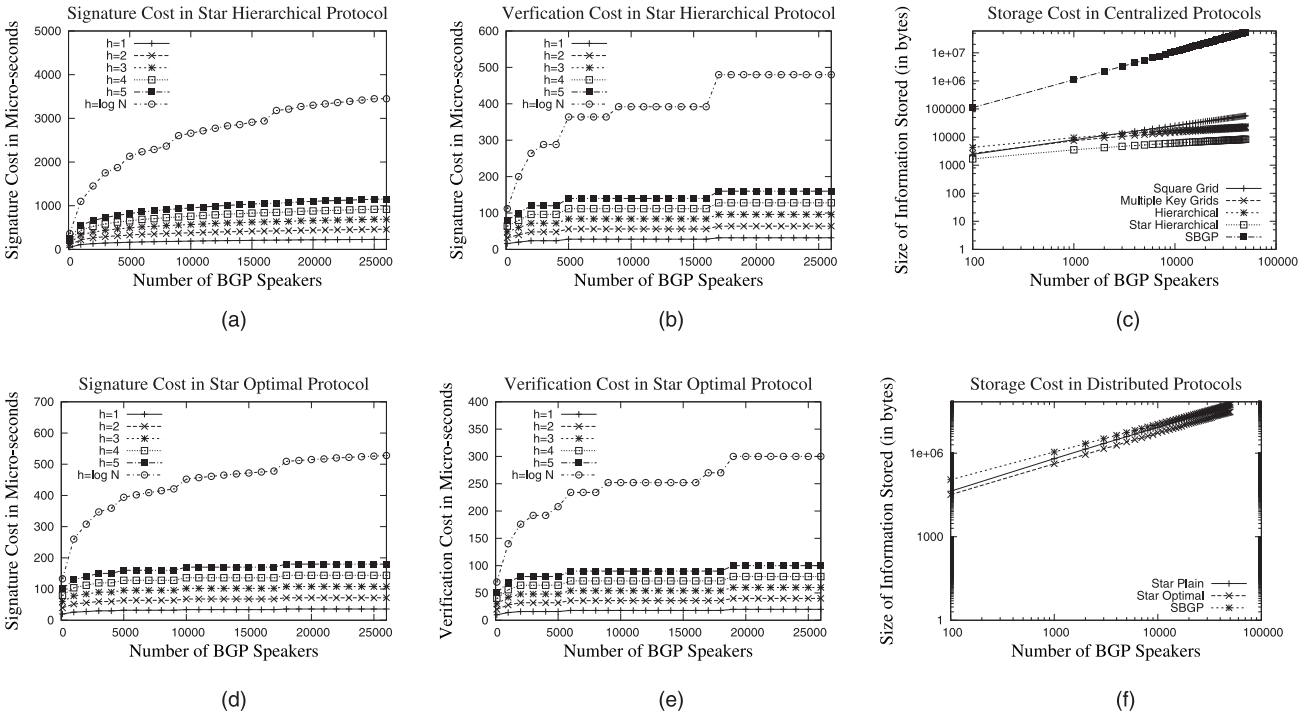


Fig. 6. Experimental results. (a) Signature cost in star hierarchical protocol for different values of h. (b) Verification cost in star hierarchical protocol for different values of h. (c) Storage overhead in global secret distribution protocols. (d) Signature cost in star optimal protocol for different values of h. (e) Verification cost in star optimal protocol for different values of h. (f) Storage overhead in local secret distribution protocols.

Optimal, which is a distributed protocol, for $h = infinity$. This implies that no node is trusted on the path and all node include their signatures for authenticating the update messages. Fig. 5c shows that the cost of signatures using the *Star optimal* protocol is lower than SBGP for small path lengths. Also, both *Star Optimal* and *Star Hierarchical* are better than SPV for signature generation. This is due to high initialization cost required in SPV. In Fig. 5f, we note that verification cost in both *Star Hierarchical* and *Star Optimal* is orders of magnitude lower than SPV and SBGP. These results show that our approach can reduce the verification cost of updates even when no BGP router is trusted.

5.4 Comparison of Signature Cost in Our Approach for Different Values of h

Finally, we illustrate the signature and verification cost for different values of h. Note that, to simplify analysis, we picked the best centralized and distributed key distribution protocol from [25] to illustrate these costs. In Figs. 6a and 6b, we show the signature and verification cost using the centralized secret distribution protocol from [25] and in Figs. 6d and 6e, we show the signature and verification cost using the distributed secret distribution protocol from [25].

We note that, an additional cost in SBGP which has not been evaluated with respect to our approaches is the cost of initial setup. In SBGP, for a sufficiently large network say 40,000 organizations and 5,000 ASes (data from [28]), the initial setup requires an exchange of about 60 MB of data. The exchange involves complex protocol message exchanges and verifications of certificates, etc. Our approaches do not involve this complexity as we do not use certificates and hence, the verification cost is nominal.

5.5 Comparison of Storage Overhead in Symmetric Key Distribution and SBGP

We evaluate the storage benefits in our protocols with respect to validating route attestations. For the symmetric key distribution protocols in our approach, we compute the storage at any BGP router by using a key length of 512 bits. For example, if a BGP router stores 225 keys, then the overall storage is given by $512 * 225$ bits. Now, for SBGP, we use the data from [28] for the signature block size and the storage required. As mentioned in [5], we consider that the overall signature block size in SBGP, which includes route and address attestations. To compute the storage requirements in SBGP, we use a simplified model. We assume that a router in SBGP stores the following certificates: the four certificates from the four Internet registries, one AS/organization certificate and one router certificate. Note that, although in SBGP the router stores other certificates as well, to simplify analysis we consider only the cost of storing these certificates. From [28], the number of bytes stored per certificate is 550. To illustrate a sample storage calculation, we note that in SBGP, each BGP router will need to store the certificates of other ASes and BGP routers. For example, if there are 100 BGP routers in the Internet then, each BGP router will need to store, $550 * (4 + 100 * 2) = 112{,}200$ bytes.

The storage overhead for these protocols is shown in Figs. 6c and 6f. The storage in centralized key distribution protocols is orders of magnitude lesser than that of SBGP. The main benefit of the centralized key distribution protocols is from reducing the storage required to validate route attestations. Specifically, in SBGP, each BGP speaker needs to look up and validate all the certificates of the nodes which are listed in the *as_path* parameter. Since route

attestations are validated at runtime this translates to increased searching and processing times. Though caching may be done for already validated paths, the storage requirements are still quite high compared to our protocols. Whereas in distributed key distribution protocols, the storage overhead is comparable with that of SBGP. However, we observe that, the storage overhead using the protocol from [25], is still much less than that of SBGP.

5.6 Comparison of Security and Performance of Symmetric Key Protocols with SBGP

SBGP uses the notions of route and address attestations to achieve six of eight requirements which are considered essential for a secure BGP protocol [5].[8] Our protocols support the route attestation feature and hence, can satisfy the three requirements relating to integrity, signature delegation, and update information timeliness. The only difference being that, in our protocols, all operations for verifying route attestations are symmetric-key-based and hence, faster and more efficient.

6 Security Analysis

In this section, we analyze the security of both the centralized and the distributed approaches proposed in this paper against the creation of invalid routes, which is the main security goal that we try to address. Our analysis focuses on the cost of adopting our protocols and the cost of breaking our protocols.

Using the centralized approach with Mittal's key protocol [24], the number of keys an AS needs to maintain is $(\log n)^2$, where n is the total numbers of ASes. Given that there are about 26,000 ASes on the Internet [27], deploying the centralized approach on the Internet requires each AS to maintain approximately 225 secret keys, which can be easily stored in memory. The verification cost for each BGP update message is $O(\log n)$. According to the centralized approach, each update message needs to contain $225 * h$ signature blocks. Considering that the current BGP message size has a limit of 4 KB, we can choose h to be three and the size of each signature block to be 6 bytes. A signature block of 6 bytes can be obtained by choosing the first 6 bytes of an MD5/SHA-1/HMAC hash. Although a 6-byte signature is not as secure as a normal MD5 hash (16 bytes) or SHA hash (20 bytes), to make an AS accept a forged signature requires forging of at least 15 $(\log n)$ such signatures, which is computationally infeasible.

In the distributed approach, we treat each AS as a center node, and all other ASes as satellite nodes for that particular AS. For $p(k, l)$, if we choose k to be 40, and l to be 4, the total number of satellite nodes that an AS can have is 91,390, which is enough as the number is ASes on the Internet is about 26,000. Each AS needs to maintain two sets of secret keys, one set consists of 40 secret keys that the AS needs to distribute to its satellite nodes, the other set consists of secret keys that other nodes distribute to this AS node, which has $26{,}000 * 4$ keys. Thus, the total number of keys an AS needs to maintain is only 104,040. Using the distributed approach, each update message needs to contain $40 * h$ signature blocks. If we choose h to be five, the 4 KB BGP message size allows each signature block to be 20 bytes, which is the output of an SHA-1 hash. Moreover, as discussed in previous paragraph, only first few bytes of hash could be used thereby increasing the value of h. Breaking the distributed approach requires the collusion of five adjacent ASes or 10 ASes, which we consider practically unlikely.

7 Discussion

Now, we briefly discuss the important issue of incremental deployment of our protocols.[9] We discuss issues like, interoperability, scalability and signature delegation, and hop-by-hop authentication, using our protocols.

7.1 Incremental Deployment

The major issues in incremental deployment are: key management, scalability, and interoperability. We note that our protocols are amenable to incremental deployment as they have the following flexibility: a node can choose not to receive the symmetric keys from the centralized manager or an individual sender and continue to operate normally. This flexibility ensures that when a node decides to use our protocols, it can do so with minimal difficulty. The same reasoning applies for the distributed protocols as well. If some node does not implement our protocol then path involving that node cannot be trusted. However, paths involving nodes that implement our protocol can be trusted. Note that, this is true for any security protocol such as SBGP and SPV.

7.1.1 Key Management: Initialization and Updates

Our protocols can leverage public-key infrastructure by using it to achieve the initial key distribution and using the symmetric protocols to achieve authentication. This approach will achieve high performance and amortize the cost of public-key infrastructure by replacing them with the relatively efficient symmetric protocols. To maintain the freshness of the symmetric keys, new keys can be periodically distributed using the initial-bootstrapping technique. We note that, there are several other approaches [30] in literature to achieve efficient key distribution when preshared keys already exist. We briefly discuss the initialization and key update techniques that can be used in our protocols.

Initialization. We assume the existence of a public-key-based infrastructure such as the one described in S-BGP [5]. However, we only assume the existence of a single public-key certificate per BGP speaker.

Initially, a sender instantiates a key distribution protocol. Now, to send an update the sender attaches the signatures with the keys from the protocol and transmits this update. An intermediate BGP speaker receiving this update message needs to use the corresponding keys to verify the message. The intermediate speaker sends a request for the keys to the originating BGP speaker and

8. For reasons of space, the requirements are not reproduced here. The reader is requested to refer to [5].

9. For a more detailed analysis of adoption of secure BGP protocols the reader can refer to [29].

includes its certificate in the request. The originating BGP speaker, upon receiving the request, verifies the certificate and uses the public-key in the received certificate to encrypt the keys that are requested by the intermediate node and transmits them accordingly.

Key updates. In our key distribution protocols, the key update process is periodic and is similar to the initialization process except for one important difference. A node may request new keys only if it sees a major change in the paths that are being advertised by the upstream routers. The keys are changed by the router generating the update but they are distributed only on demand. If there is no change in the path within the system parameter h then an intermediate node may choose not to verify the signatures on the update. By doing so the intermediate node does not need to send a request for the new keys. On the other hand, if the paths being advertised change, then the intermediate node requests new keys before taking a decision on the new path.

7.1.2 Scalability

Our protocols are scalable in terms of storage required at the users and the number of signatures per message. By using our centralized protocols the BGP speakers store $O(\log^2 N)$ keys and using the distributed approach requires a storage of $O(N \log N)$ keys where N is the number of BGP speakers. Also, note that, these numbers are relatively quite low when compared to the sizes of the certificates that are stored as part of the SBGP protocol. Addition of new BGP speakers can be handled in our protocols by using a larger pool of keys during the initial distribution. The additional keys can be given to the new BGP speakers who join later. We also point out that issues such as certificate distribution and revocation exist in public-key infrastructure and are no different in scale to our protocols although the frequency of these tasks may be relatively low.

7.1.3 Interoperability

Our protocols can interoperate with existing security mechanisms for BGP. For example, the top-level public-key infrastructure in SBGP can be used to achieve the key distribution for our protocols. Also, SBGP may be used between BGP speakers when the one of the nodes is unwilling to implement our protocols. For example, in our distributed protocol, if nodes are not willing to receive keys from the sender, they can continue using public-keys or SPV to sign the update messages. This approach will not affect the working of the existing mechanisms.

7.1.4 Signature Delegation

We use signature delegations, similar to those proposed in SBGP [5]. To generate a signature delegation, a sender includes the AS number of the recipient node along with the Update message and signs this content. This prevents attacks wherein an arbitrary node inserts itself into the ASPATH and propagates a longer ASPATH for the advertised prefix.

7.2 Hop-by-Hop Authentication

Our protocols can also be leveraged to achieve hop-by-hop authentication. The approach is as follows, each node computes an XOR of the common keys held by the next-hop neighbor and generates an additional signature using this value. The next-hop neighbor uses this value to confirm the authenticity of the sender. We note that, authentication between hops can be very effective against spoofing attacks.

8 Conclusion

We make three key contributions in this paper. First, we show that the right trade-off between efficiency and security for BGP could be achieved by adding the little bit of trust on BGP routers. We present a new flexible threat model where for any path of length k, at least one BGP router is trustworthy. Second, we present two new symmetric key approaches to securing BGP: the centralized key distribution approach and the distributed key distribution approach. Third, we evaluated the efficiency of the two approaches with previous approaches to securing BGP. The evaluation results show that our approaches are significantly more efficient than previous approaches. Also, we have discussed the deployment issues and important concerns like key management and interoperability to illustrate the feasibility of our protocols.

Acknowledgments

The authors thank the editor and the anonymous reviewers for their constructive comments. The work of Alex X. Liu is supported in part by the US National Science Foundation (NSF) under Grant Numbers CNS-0716407, CNS-0916044, and CNS-0845513. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the US National Science Foundation. The preliminary version of this paper titled "Symmetric Key Approaches to Securing BGP—A Little Bit Trust is Enough" was published in ESORICS 2008 [1]. Alex X. Liu is the corresponding author of this paper.

References

[1] B. Bruhadeshwar, S.S. Kulkarni, and A.X. Liu, "Symmetric Key Approaches to Securing BGP—A Little Bit Trust Is Enough," *Proc. 12th European Symp. Research Computer Security (ESORICS)*, Oct. 2008.

[2] Y. Rekhter and T. Li, "A Border Gateway Protocol 4 (BGP 4)," IETF RFC 1771, 1995.

[3] L. Subramanian, V. Roth, I. Stoica, S. Shenker, and R. Katz, "Listen and Whisper: Security Mechanisms for BGP," *Proc. First Symp. Networked Systems Design and Implementation (NSDI '04)*, 2004.

[4] G. Goodell, W. Aiello, T. Griffin, J. Ioannidis, P. McDaniel, and A. Rubin, "Working around BGP: An Incremental Approach to Improving Security and Accuracy of Inter-Domain Routing," *Proc. Network and Distributed Systems Security (NDSS)*, pp. 75-85, 2003.

[5] S. Kent, C. Lynn, and K. Seo, "Secure Border Gateway Protocol," *IEEE J. Selected Areas in Comm.*, vol. 18, no. 4, pp. 582-592, Apr. 2000.

[6] R. White, "Securing BGP through Secure Origin BGP," *The Internet Protocol J.*, vol. 6, no. 3, pp. 15-22, 2004.

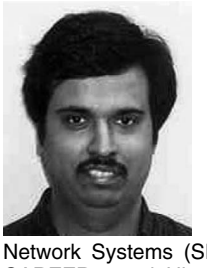
[7] P.C. Van Oorschot, T. Wan, and E. Kranakis, "On Interdomain Routing Security and Pretty Secure BGP (psBGP)," *ACM Trans. Information and System Security*, vol. 10, no. 3, July 2007.

[8] B.R. Smith and J.J. Garcia-Luna-Aceves, "Securing the Border Gateway Routing Protocol," *Computer Comm.*, vol. 21, no. 3, pp. 203-210, 1998.

[9] Y. Hu, A. Perrig, and D. Johnson, "Efficient Security Mechanisms for Routing Protocols," *Proc. Network and Distributed Systems Security (NDSS)*, 2003.

[10] Y. Hu, A. Perrig, and M. Sirbu, "SPV: Secure Path Vector Routing for Securing BGP," *Proc. ACM SIGCOMM '04,* 2004.
[11] R. Mahajan, D. Wetherall, and T. Anderson, "Understanding BGP Misconfiguration," *Proc. ACM SIGCOMM '02,* Oct. 2002.
[12] K. Butler, P. McDaniel, and W. Aiello, "Optimizing BGP Security by Exploiting Path Stability," *Proc. ACM Conf. Computer and Comm. Security (CCS '06),* Oct./Nov. 2006.
[13] D. Nicol, S. Smith, and M. Zhao, "Efficient Security for BGP Route Announcements," Technical Report TR-2003-440, Dartmouth Univ., 2002.
[14] M. Zhao, S. Smith, and D. Nicol, "Evaluating the Performance Impact of PKI on BGP Security," *Proc. Fourth Ann. PKI Research Workshop (PKI '05),* 2005.
[15] http://bgpupdates.potaroo.net/instability/bgpupd.html, 2011.
[16] S. Kent and R. Atkinson, "Security Architecture for the Internet Protocol," IETF RFC 2401, 1998.
[17] P. McDaniel, K. Butler, T. Farley, and J. Rexford, "A Survey of BGP Security Issues and Solutions," *Proc. IEEE,* vol. 98, no. 1, pp. 100-122, Jan. 2010.
[18] R.C. Merkle, "Protocols for Public Key Cryptosystems," *Proc. IEEE Symp. Security and Privacy,* 1980.
[19] R. Merkle, "A Digital Signature Based on a Conventional Encryption Function," *Proc. Conf. Theory and Applications of Cryptographic Techniques on Advances in Cryptology (CRYPTO '87),* pp. 369-378, Oct. 1987.
[20] E. Wong, P. Balasubramanian, L. Alvisi, and V. Shmatikov, "Truth in Advertising: Lightweight Verification of Route Integrity," *Proc. 26th ACM Ann. Symp. Principles of Distributed Computing (PODC '07),* pp. 147-156, 2007.
[21] X. Hu and Z.M. Mao, "Accurate Real-Time Identification of IP Prefix Hijacking," *Proc. IEEE Symp. Security and Privacy,* pp. 3-17, May 2007.
[22] S.S. Kulkarni, M.G. Gouda, and A. Arora, "Secret Instantiation in Ad-Hoc Networks," *Computer Comm.,* vol. 29, pp. 200-215, 2006.
[23] A.S. Aiyer, A. Lorenzo, and M.G. Gouda, "Key Grids: A Protocol Family for Assigning Symmetric Keys," *Proc. IEEE Int'l Conf. Network Protocols,* 2006.
[24] N. Mittal, "Space-Efficient Keying in Wireless Communication Networks," Technical Report UTDCS-26-07, Dept. of Computer Science, Univ. of Texas at Dallas, 2007.
[25] B. Bruhadeshwar and S. Kulkarni, "An Optimal Symmetric Secret Distribution for Star Networks," Technical Report MSU-CSE-07-196, Michigan State Univ., 2007.
[26] M.G. Gouda, S.S. Kulkarni, and E.S. Elmallah, "Logarithmic Keying of Communication Networks," *Proc. Eighth Int'l Symp. Stabilization, Safety, and Security of Distributed Systems (SSS '06),* 2006.
[27] "Cidr Report for Third November 2007," http://www.cidr-report.org/as2.0/, 2007.
[28] S. Kent, C. Lynn, J. Mikkelson, and K. Seo, "Secure Border Gateway Protocol (S-BGP) Real World Performance and Deployment Issues," *Proc. Symp. Network and Distributed Systems Security (NDSS),* 2000.
[29] H. Chan, D. Dash, A. Perrig, and H. Zhang, "Modeling Adoptability of Secure BGP Protocol," *Proc. SIGCOMM,* pp. 279-290, 2006.
[30] L. Eschenauer and V.D. Gligor, "A Key-Management Scheme for Distributed Sensor Networks," *Proc. ACM Conf. Computer and Comm. Security (CCS '02),* 2002.

Bezawada Bruhadeshwar received the PhD degree in computer science and engineering from Michigan State University, in 2005. He is currently working as an assistant professor at the International Institute of Information Technology, Hyderabad, India. His research interests are in security, networking, and dependable systems. He is a member of the IEEE.



Sandeep S. Kulkarni received the BTech degree in computer science and engineering from the Indian Institute of Technology, Mumbai, India, in 1993, and the MS and PhD degrees in computer and information science from Ohio State University, Columbus, in 1994 and 1999, respectively. Since 1999, he has been working at Michigan State University, East Lansing, where he is currently an associate professor. He is a member of the Software Engineering and Network Systems (SENS) Laboratory. He is a recipient of the NSF CAREER award. His research interests include fault-tolerance, security, automated program synthesis, distributed systems, group communication, and self-stabilization. He is a member of the IEEE.

Alex X. Liu received the PhD degree in computer science from the University of Texas at Austin in 2006. He is currently an assistant professor in the Department of Computer Science and Engineering at Michigan State University. He received the IEEE & IFIP William C. Carter Award in 2004 and the US National Science Foundation (NSF) CAREER award in 2009. He received the MSU College of Engineering Withrow Distinguished Scholar Award in 2011. His research interests focus on networking, security, and dependable systems.