Published: 25 November 2024

RESEARCH-ARTICLE

The Multiple Benefits of a Secure Transport for BGP

THOMAS WIRTGEN

NICOLAS RYBOWSKI

CRISTEL PELSSER

OLIVIER BONAVENTURE

Proceedings of the ACM on Networking, Volume 2, Issue CoNEXT4 (December 2024)
https://doi.org/10.1145/3696406
EISSN: 2834-5509

The Multiple Benefits of a Secure Transport for BGP

THOMAS WIRTGEN, ICTEAM, UCLouvain, Belgium and WEL Research Institute, Belgium
NICOLAS RYBOWSKI, ICTEAM, UCLouvain, Belgium and WEL Research Institute, Belgium
CRISTEL PELSSER, ICTEAM, UCLouvain, Belgium
OLIVIER BONAVENTURE, ICTEAM, UCLouvain, Belgium and WEL Research Institute, Belgium

BGP distributes prefixes advertised by Autonomous Systems (ASes) and computes the best paths between them. It is the only routing protocol used to exchange interdomain routes on the Internet. Since its original definition in the late 1980s, BGP uses TCP. To prevent attacks, BGP has been extended with features such as TCP-MD5, TCP-AO, GTSM and data-plane filters. However, these ad hoc solutions were introduced gradually as the Internet grew. In parallel, TLS was standardized to secure end-to-end data-plane communications. Today, a large proportion of the Internet traffic is secured using TLS. Surprisingly, BGP still does not use TLS despite its adequate security features to establish BGP sessions. In this paper, we make the case for using a *secure transport* with BGP. This can be achieved with TLS combined with TCP-AO or by replacing TCP by QUIC. This protects the BGP stream using established secure transport protocols. In addition, we show that a secure transport using X.509 certificates enables BGP routers to be securely and automatically configured from these certificates. We extend the open-source `BIRD` BGP daemon to support TLS with TCP-AO and QUIC, to handle such certificates and demonstrate several use cases that benefit from the secure and automated capabilities enabled by our proposal.

CCS Concepts: • **Networks** → **Network protocol design**; **Routing protocols**; *Routers*; • **Security and privacy** → **Network security**.

Additional Key Words and Phrases: BGP, TLS, QUIC, X.509 Certificates, Certificates, Network Automation

ACM Reference Format:
Thomas Wirtgen, Nicolas Rybowski, Cristel Pelsser, and Olivier Bonaventure. 2024. The Multiple Benefits of a Secure Transport for BGP. *Proc. ACM Netw.* 2, CoNEXT4, Article 36 (December 2024), 23 pages. https://doi.org/10.1145/3696406

1 Introduction

The Internet protocols were originally designed without strong security requirements. Remote terminal access was done using `telnet` or `rsh` over plaintext TCP connections, which were vulnerable to eavesdropping and other types of attacks until the development of `ssh` in the late 1990s [101]. File transfers using the FTP protocol or email using SMTP, POP or IMAP also relied on regular TCP connections. These TCP applications gradually adopted TLS [33, 70] after the release of the first version of the Transport Layer Security specification [3]. The same was true for HTTP, which adopted TLS in May 2000 [76]. Today, measurements show that only a tiny fraction of websites

Authors' Contact Information: Thomas Wirtgen, thomas.wirtgen@uclouvain.be, ICTEAM, UCLouvain, Louvain-la-Neuve, Belgium and WEL Research Institute, Wavre, Belgium; Nicolas Rybowski, nicolas.rybowski@uclouvain.be, ICTEAM, UCLouvain, Louvain-la-Neuve, Belgium and WEL Research Institute, Wavre, Belgium; Cristel Pelsser, ICTEAM, UCLouvain, Louvain-la-Neuve, Belgium, cristel.pelsser@uclouvain.be; Olivier Bonaventure, ICTEAM, UCLouvain, Louvain-la-Neuve, Belgium and WEL Research Institute, Wavre, Belgium, olivier.bonaventure@uclouvain.be.

ACM 2834-5509/2024/12-ART36
https://doi.org/10.1145/3696406

does not support HTTPS [30]. This is largely due to the Let's Encrypt initiative, which provides free TLS certificates [1], making TLS the default for Internet communications. Recently, the IETF standardized the QUIC protocol [49], which provides stronger protection than TCP over TLS. QUIC has already been adopted by several application-level protocols including HTTP/3 [11] or DNS [45].

Internet routing protocols are less vulnerable to eavesdropping than application layer protocols such as HTTP or SMTP because they are mainly used on controlled point-to-point links. Hence, they faced fewer incentives to adopt solutions to encrypt the information exchanged between routers. The IETF routing working groups have mainly focused on authenticating the routing information exchanged between routers, using hash-based techniques to authenticate packets [28, 40, 56]. Current best practices for securing the transport of the Border Gateway Protocol (BGP) combine these per-packet authentication schemes [40, 95] with various data-plane filters [26] and the Generalized TTL Security Mechanism (GTSM) [73]. From an operational point of view, the main difficulty with hash-based packet authentication techniques is the distribution of the required authentication keys. Ideally, these keys should be changed periodically for security reasons [8]. In practice, they are usually manually configured on the routers and rarely changed, especially for sessions between routers managed by different operators.

Researchers proposed many techniques to secure interdomain routing. Detailed surveys [18, 48] have compared different approaches. S-BGP [54], one of the first extension to secure BGP has been very influential to improve interdomain routing security. It assumed that BGP sessions would be established over IPSec tunnels to prevent packet injection attacks and extended BGP to enable ASes to sign their BGP Updates. Unfortunately, when S-BGP was proposed in the late 1990s, it required too much CPU and memory [55]. Operators and vendors did not consider it as a deployable approach. The ideas proposed in S-BGP influenced the design of BGPsec [58] and the Resource Public Key Infrastructure (RPKI) [57]. It is currently used to cryptographically sign a prefix to ASes that are authorized to advertise it. However, as of May 2024, its deployment is still slow, with 50% of Internet prefixes that are still not protected [25]. Other extensions to RPKI are currently being discussed within the IETF, such as ASPA [6].

Thanks to the widespread adoption of TLS by servers, CPU vendors have added dedicated instructions to directly support encryption and authentication with very little impact on CPU performance. In this paper, we explore the benefits brought by **BGP over a *secure transport* (BGPoST)** such as TLS and QUIC. This paper complements the ongoing efforts to secure interdomain routing with the RPKI and later BGPsec. While RPKI focuses on securing Internet resources, we propose to secure the transport of BGP messages, that is, not the routing itself. We do not propose any changes to the BGP protocol itself, making our approach simpler to adopt and deploy.

Contributions. This paper is articulated in two main contributions. The first contribution is the use of a secure transport protocol for BGP, which we motivate in Section 2. We implement both BGP over QUIC, and BGP over TLS on top of the `BIRD` open-source routing daemon (Section 3). Our experiments show that BGPoST works and does not strongly impact BGP routers. Second, we propose to extend X.509 certificates to autoconfigure routers. We design an architecture that allows a provider to issue X.509 certificates and deploy them to its clients (Section 4). We present two use cases that are directly enabled by BGPoST certificates. First, we implemented a mechanism to dynamically fall back to a tunnel pre-established using BGPoST certificates when the connectivity of one of the providers of a multi-homed stub fails. Second, we show that leveraging BGPoST and BGPoST certificates can improve blackholing services (Section 5).

We end this paper by presenting some future directions that our approach opens for BGPoST and BGPoST certificates (Section 6) and by concluding this paper (Section 8). The Appendix A provides two additional use cases enabled by X.509 certificates. The first (A.1) enables a customer

to dynamically adjust QoS rules on the provider side. The second (A.2) simplifies the configuration of inter-AS anycast services.

Ethical considerations. This work does not raise any ethical concerns.

2 Securing BGP with TLS over TCP, or QUIC

The initial version of BGP [61] allowed for the use of any reliable transport protocol to exchange messages. However, due to the prevalence of TCP support in routers at the time, TCP became the de facto standard transport protocol for BGP. Subsequent versions of the protocol, including the latest BGP4 version, continue to use TCP. As discussed in the previous section, TCP only no longer satisfies the requirements of today's networks. Remote peering is gaining popularity for connecting to cloud providers and Internet eXchange Points (IXPs) [37, 63]. This method allows routers to establish BGP sessions over the Internet with these entities. In practice, a remote peering reseller provides the Layer 2 or Layer 3 network required to remotely access these sites. However, the BGP session with the cloud provider or IXP is not secure as it is transiting through the reseller's network. Although IPsec has been used by network operators since the early 2000s, a survey [99] we carried out in the summer of 2024 showed that only 6% of the operators who took part in the survey were deploying BGP over IPsec. To increase the security of BGP traffic, it is reasonable to use protocols such as TLS/TCP or QUIC. These protocols can provide the encryption and authentication capabilities required to protect control-plane data. This section discusses the advantages of using a secure transport protocol for BGP and the mechanisms required to support such a change.

Authenticating BGP routers. Authenticating and authorizing routers, and by extension, associated BGP sessions, is a two-fold process. First, a router must be configured to accept BGP sessions from authorized peer routers only. Second, a router must ensure that subsequent messages sent over the authorized BGP session are still originated by their authorized peer router.

Currently, routers authenticate peer routers by their IP addresses. Hence, network operators must configure their routers with Access Control Lists (ACLs) containing the IP addresses of the authorized remote BGP routers. Any connection from an unauthorized address will be rejected. BGP messages are then authenticated thanks to hash-based techniques such as TCP-MD5 [40] or TCP-AO [95]. Furthermore, many BGP implementations allow dynamic BGP sessions. This feature enables the router to listen for incoming BGP connections on an IP prefix instead of a unique IP address. Once the remote BGP router has established the connection, the BGP router configures the incoming BGP session on the fly. Compared to establishing a traditional BGP session, this approach reduces the router's configuration when peers change frequently. However, the BGP router must trust the remote router as it has no way of identifying it in the prefix. This technique is safe in a controlled environment, e.g., in the same local area network (LAN) of a single AS, where foreign routers can hardly be introduced.

There are two types of TLS certificates used with TCP and QUIC: the classical server certificates used by web servers and the client certificates. The latter, while optional, are typically used to provide mutual authentication (mTLS) [78]. This scheme is often used to access governmental services or banking services [72]. We leverage such certificates to ensure the mutual authentication of BGP routers. In our approach, each router is provisioned with a unique certificate assigned by the network operators. Then, each router can be configured with an ACL based on TLS certificates of authorized BGP neighbors. The trust in certificates can also come from a Public Key Infrastructure (PKI) [13] that ensures their validity. In both cases, this guarantees a more secure authentication than the ACLs based on IP addresses or prefixes that can be spoofed. By using TLS certificates, dynamic peering can be extended beyond the AS boundaries by including certificates from specific ASes or dedicated routers in the router's ACL. Note that these certificates do not necessarily need to be provided by an independent certification authority. For iBGP sessions, the certificates are

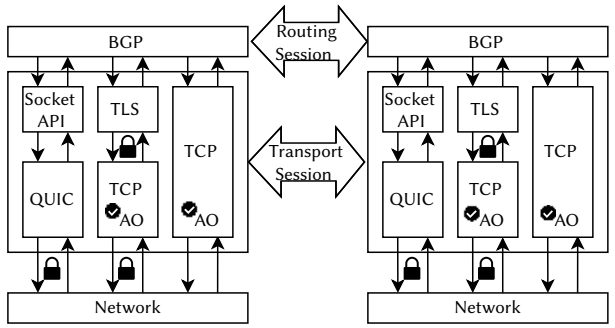


Fig. 1. BGP can interact with different transport protocols to send messages. Our approach supports TCP (with TCP-AO), TCP/TLS (with TCP-AO), or QUIC. The padlock is the location where BGP data is encrypted, and "AO" is the TCP-AO option used to authenticate TCP segments.

used by routers managed by a single network operator and a locally-maintained PKI is sufficient. For eBGP sessions, we propose a mechanism to distribute certificates described in Section 4.

Countering BGP attacks with spoofed packets. BGP is vulnerable to attacks that force the termination of a TCP connection [67]. Malicious data can also be injected to corrupt the BGP session. With BGP over TLS, the session is still vulnerable to TCP-RST or FIN attacks as the TCP header is not encrypted within the TLS header. Hence, the underlying TCP session must be secured with TCP-AO or TCP-MD5. This requires adding a Message Authentication Code (MAC) field to the TCP header to authenticate the TCP segment. The password can be negotiated manually by the network operators or derived from the TLS handshake, as suggested by a recent IETF draft [74].

With BGP over QUIC, these attacks become almost impossible. To inject a packet in a QUIC connection, an attacker needs to predict the current QUIC connection identifier. This is feasible if the attacker is able to observe the connection packets, but peers can mitigate the attack by regularly changing their connection identifiers [49]. A packet with an invalid connection identifier is simply dropped. The next step for the attacker is to predict the packet number which is part of the encrypted header. An attacker cannot easily determine this number even by capturing packets. To inject data the attacker would then need to predict the encryption and authentication keys that were negotiated using TLS 1.3 during connection establishment [77, 94]. Injecting QUIC packets is thus much more difficult than injecting TCP packets. Furthermore, routers can still use techniques such as access lists and GTSM [73] to restrict the packets that reach routers.

Figure 1 illustrates our unified architecture for the BGP protocol to interact with secure transports. The QUIC protocol handles the authentication, encryption, and transport of BGP messages. In contrast, when using TLS over TCP, the TCP layer manages the transport of BGP data and the authentication of the TCP header with TCP-AO or TCP-MD5. The TLS layer handles the encryption and authentication of BGP messages. In the figure, the padlocks are the location where BGP data is encrypted, and the check mark indicates the authentication of TCP segments using TCP AO.

Summary. As detailed in this section, many techniques have been developed over time to improve the security of BGP over a TCP connection. Table 1 summarizes the main security measures used to secure the TCP session. Leveraging QUIC or TLS over TCP to achieve BGPoST efficiently replaces all these techniques as they are “built-in” such secure transport sessions.

3 Prototyping BGPoST

Internet routing protocols are currently tightly coupled to their underlying transport protocols. For example, BGP has been standardized to use TCP. We argue that Internet routing protocols should

Security Feature	Existing methods	QUIC	TLS/TCP
Enforcing 1-Hop sessions on adjacent links	GTSM	☑	☑
Router authentication	Pre-configuring BGP sessions by router IP addresses	☑	☑
Preventing BGP packets spoofing	TCP-MD5 or TCP-AO	☑	☑ (w/ TCP-AO)
Protection of BGP stream (1-Hop or MultiHop)	IPsec (or any encrypted tunnel)	☑	☑

Table 1. Summary of current BGP security techniques built directly into QUIC or TLS/TCP.

be less dependent on the transport layer in the future and should negotiate the use of different secure transport protocols. We modified the `BIRD` routing daemon (v2.0.10 for BGP over QUIC & v2.14 for BGP over TLS/TCP) to allow network operators to select the transport protocol when establishing a BGP session. BGP can now send messages using TCP, TLS over TCP, or QUIC.

The remainder of this section further describes the integration of TLS over TCP and QUIC into BGP. We conclude by investigating the performance impact of replacing TCP with a secure transport in Section 3.3.

3.1 Securing BGP with TLS and TCP-AO

Integrating a TLS library into a TCP application is often straightforward. This is also the case for BGP. Instead of exchanging data directly from the TCP socket to the BGP session, we first pass it through a TLS library. Our implementation uses the `picotls` library [91], which implements TLS 1.3. The integration is simple: whenever BGP needs to send data, we encrypt it first using `picotls`. This encrypted data is then written to the TCP socket. To receive data, the process is similar: TCP delivers it to the `picotls` library which performs decryption and passes the result to BGP.

Adding TLS into `BIRD` required a few modifications to the code due to its cross-platform architecture that can run on various operating systems, including Linux and BSD variants. `BIRD` has an abstraction layer over the OS socket interface. This layer provides an API for all types of sockets. When a routing protocol in `BIRD` requires a TCP connection, it uses the dedicated `BIRD` TCP socket API to read and write data. Similarly, we integrated TLS into `BIRD` by adding a TLS abstraction layer, enabling any routing protocol in `BIRD` to leverage TLS.

This integration is the first step towards securing BGP. However, TLS is still vulnerable to the injection of corrupted TCP packets into the connection. When a TCP connection has been altered such that the decryption of a TLS record fails, the TLS session is terminated with an alert. In addition, when TCP RST packets are received, the session fails and BGP routes are removed because the BGP session is disconnected. To prevent these attacks, `BIRD` already uses TCP-MD5 to authenticate the entire TCP segment. One may leverage this authentication technique to secure the TLS session. However, this approach is vulnerable due to the depreciation of the MD5 hash algorithm [97]. TCP-AO is available since version 6.7 of the Linux kernel, offering more secure authentication algorithms and more sophisticated key management than TCP-MD5 [92].

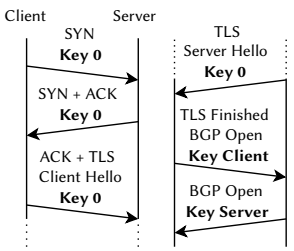


Fig. 2. Sequence diagram of TCP-AO keys negotiation derived from the TLS handshake.

We modified `BIRD` to support TCP-AO negotiation for BGP. Our implementation supports two modes. The first mode is the simplest, it allows the two BGP routers to use a pre-shared key that must be manually added, just as the TCP-MD5 BGP implementation does. The second mode, described in a recent IETF draft [74], enables automatic negotiation of TCP-AO keys. By taking advantage of the TLS session, any implementation of this draft enables TCP-AO keys to be derived from the secret established by TLS.

The principle is illustrated in Figure 2. When the TCP client sends the SYN message, the TCP segment is signed with a default TCP-AO key. If the server responds with an authenticated SYN+ACK with the same key, this means that both endpoints support TCP-AO. From this point on, the TLS handshake takes place. When the TLS handshake is complete, routers derive the TCP-AO keys to be used. In total, two symmetric keys are used. The first is used by the client to authenticate its messages, and the second is used by the server. The server checks that the messages have been authenticated with the "client" key and the client does the same, checking that the TCP segment has been authenticated with the "server" key. Once the TLS handshake is complete, the rest of the BGP conversation is authenticated by TCP-AO.

3.2 Securing BGP with QUIC

QUIC [49] is a protocol that runs on top of UDP. Currently, a large majority of QUIC implementations are user-space libraries that applications can use to exchange data over the network. For our prototype, we opted for the `picoquic` implementation [44]. It includes a broad range of features defined by the QUIC standard [49] and related extensions. Given its extensive range of features, `picoquic` presents a suitable choice for researchers and developers willing to experiment with the latest extensions of the QUIC protocol.

Integrating `picoquic` into `BIRD` is challenging due to the asynchronous nature of the `picoquic` API. When data is passed to the `picoquic` API, a callback function is required to handle the event after the data has been received and decrypted by the QUIC library. Integrating this library directly into `BIRD` would require a redesign of the `BIRD` I/O loop. Currently, the `BIRD` I/O loop uses the `BIRD` abstraction layer but keeps a traditional "socket" approach, using functions such as `write`, `read` and `listen`. This architecture is not compatible with callback functions as required by `picoquic`. To simplify the interaction between `BIRD` and `picoquic`, we created a socket API overlay on top of `picoquic` that fits this socket approach while also providing a uniform interface to QUIC libraries. As QUIC libraries have custom APIs, which can vary from one implementation to another, changing QUIC implementations usually requires rewriting the application code to adapt to these differences. However, with our approach, implementing the socket API overlay for the new QUIC library is sufficient to integrate it seamlessly into routing stacks. Our QUIC socket API is composed of 4737 lines of C code (LoCs) and supports `picoquic` [44] and `msquic` [64] partially.

We integrated the socket API into the `BIRD` [22] routing stack, thereby making it available to all the routing protocols that the stack supports. Embedding our QUIC socket API is easy and only requires adding a few lines of code to `BIRD` as it follows the BSD socket interface approach. Abstracting QUIC through a socket API is a deliberate decision to minimize modifications to the `BIRD` code base, with 759 LoCs required to support our QUIC socket API. Recent research showcased the flexibility of this socket API by porting OSPF over QUIC [82], then leveraging QUIC features with OSPF.

3.3 Performance Considerations

We evaluate the performance of our BGPoST implementations with a specific experimental setup. Our evaluation does not consider an in-depth analysis on how our implementation performs but rather give confidence in the soundness of our new approach.

First, to build network topologies, we use emulation to limit the number of physical machines deployed per experiment. To minimize the measurement noise and other disturbances that can be generated by emulation on a single machine, we exploit the following features offered by the Linux kernel. First, we create a new network namespace (`netns`) per emulated router to ensure that each node has its own network stack. Second, we isolate each emulated node on its own dedicated core on which we pin a `BIRD` process. With the kernel arguments `isolcpus` and `nohz_full` [93] we

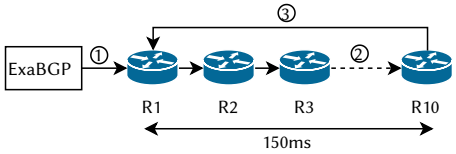


Fig. 3. Line of 10 routers used to measure BGP Update propagation times. Only four routers are shown.

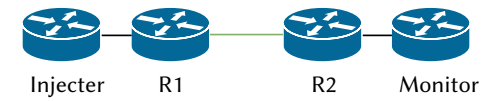


Fig. 4. Topology used to measure the overhead of the transport under load.

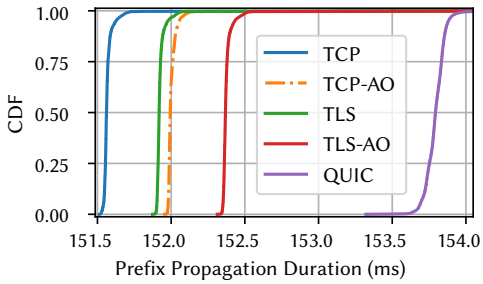


Fig. 5. Distribution of the time required to send one BGP Update every 50 ms according to the transport used in the topology described in Fig. 3. In total, 30 000 BGP updates are sent during the experiment.

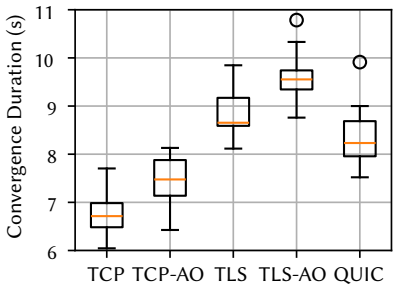


Fig. 6. Convergence Time of BGP using several transport between R1 and R2 over 15 runs.

force the kernel scheduler to use other cores to run user-space processes. Finally, we use virtual Ethernet pairs to communicate between the different network namespaces. Bandwidth and latency are configured on links with `tc`, for which we set a 15 ms delay and a bandwidth of 1 Gbps.

The Linux kernel version we are using is v6.7.0, running Debian 12 on a server with four AMD Opteron™ 6176 2.3GHz CPUs and 48 GB RAM. We deactivated the Simultaneous Multithreading (SMT) to limit process competition on a given CPU core.

BGP route propagation time. We evaluate the time required to propagate a prefix in the topology composed of a line of 10 routers, as depicted in Figure 3. Routes are advertised every 50 ms by an `ExaBGP` node, which is directly connected without delay to R1. The methodology is as follows: `ExaBGP` announces a route to R1 (1), R1 propagates the route to R2, and so on up to R10 (2). Finally, R10 sends the route back to R1 (3). We then calculate the time elapsed between the moment the route is received from ExaBGP on router R1 and the moment it is re-announced by R10 to R1. This gives an estimate of the propagation time required per route. Figure 5 illustrates this time distribution as a function of the transport layer used.

For the baseline configuration, BGP uses a TCP session on all routers in the topology. In this scenario, the median prefix propagation duration is 151.6 ms. Since this particular topology has a fixed delay of 150 ms, propagating a BGP update message through the ten routers takes a median of 1.6 ms. When TLS is used, the propagation time slightly increases to 151.9 ms because of the encryption of BGP messages. Authenticating TCP segments with the TCP-AO option also increases the propagation time of BGP messages. However, this is rather negligible as it adds a median of 0.4 ms to the total propagation time when using plain TCP or TLS over TCP to compute and verify the signature. When propagating routes over BGP sessions using QUIC, the added propagation overhead increases more significantly while the total time remains acceptable, reaching 153.8 ms. There are several explanations for this increase. First, QUIC implementations are not yet as optimized

as TCP implementations. Second, as QUIC is a user-space protocol, all transport logic is located in user-space hence increasing the number of context-switches during data processing, while TCP and TCP-AO mostly happens in the kernel. Third, our integration of the socket API is not optimized yet, inducing many copies of the data made before reaching `BIRD` for BGP message processing.

BGP convergence time. The previous experiment demonstrated the performance of BGP with multiple transport layers under normal operating conditions, i.e., when BGP processes a small number of messages per second, as it would do on the Internet. Now, we evaluate how a router reacts during a cold start by feeding it with a full routing table. We verify that it can handle such a large number of routes, although this is a rare event in the life of a router.

To this end, we create a network topology (illustrated in Figure 4) with four routers named Injecter, R1, R2, and Monitor. The Injecter is fed with a full routing table from a RIPE RIS snapshot of `rrc01` dated on the 15th of November 2023 at 11.15 AM [81]. The Injecter selects the best routes from `rrc01` before sending them over a BGP-TCP session to R1. In total, 991k IPv4 routes and 200k IPv6 routes are sent to R1. Before the experiment, all routers run BGP. The Injecter (running GoBGP) has been fed with the RIS snapshot. All BGP sessions are set up except the one between the Injecter and R1. We wait for the other sessions to converge and the Injecter to load the RIS snapshot. Then we start the BGP session between the Injecter and R1. The Injecter sends routes to R1. When R1 receives the routes from the Injecter, it propagates them to R2, and then R2 sends the routes to the Monitor. The Monitor is connected to R2 through a BGP-TCP session and logs all BGP update messages received. Routers R1, R2, are configured to establish a different transport depending on the experiment.

We measure the convergence time under stress, i.e., the time needed to propagate the full routing table from the Injecter to the Monitor. Figure 6 reports the time required to process the full routing table coming from the Injecter. As expected, the time required to converge upon the transmission of a full feed is larger than the propagation of routes in isolation. When looking at BGP secured by TCP-AO, TLS and their combination, the convergence time increases due to the processing required to protect the BGP messages; However, we argue that the scale of this increase does not significantly affect the functioning of a BGP router. The convergence time when using BGP over QUIC is slightly better than over TLS. One possible explanation lies in the code architecture. In the case of QUIC, the socket API is designed to collect several BGP messages that will be encrypted in one API call. In contrast, in our TLS implementation, when `BIRD` generates a BGP message, it is immediately encrypted by `picotls` without any attempt to batch multiple messages before calling the encryption API. As a result, this API is called more frequently for `picotls` than for QUIC. An analysis using the Linux `perf` tool showed that the TLS implementation spend around 50% more CPU cycles than the QUIC implementation within the cryptographic API.

BGP is ready to adopt secure transport protocols. Our experiments demonstrate that using a secure transport layer does not significantly increase the propagation and convergence times of BGP compared to a classic TCP session. We argue this is an acceptable cost given the additional security benefits provided by QUIC and TLS over TCP. When BGP was first used, akin to HTTP, SMTP, IMAP, and FTP, security was not a main concern. However, over time, all these protocols except BGP have been modified to support an additional protection layer, despite its impact on performance. While improving our prototypes could alleviate some of the overhead we measured, we argue that the use of TLS within Internet protocols is considered acceptable from a performance viewpoint. As the adoption of TLS with HTTP has increased without major performance concerns, we expect the same for BGP over TLS or QUIC, especially since most routers have now hardware with cryptographic instructions offload to dedicated hardware [21].

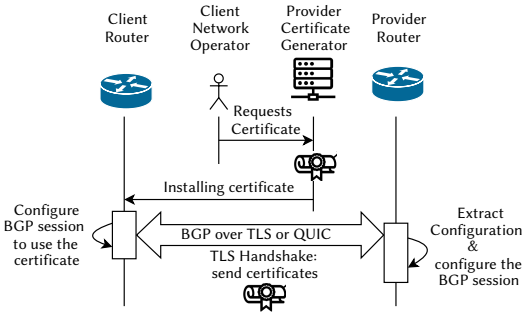


Fig. 7. A network operator can request an X.509 certificate from its provider. It is automatically generated with the client options. The client operator installs it on its router. The router then establishes a BGP session over QUIC or TLS. The certificate is sent to the provider router, which extracts the configuration it contains and applies it to this BGP session.

```
Issuer: C = Country, ST = State,
    L = City, O = Organization
Signature Algorithm: ED25519
....
X509v3 extensions:
 [Some fields omitted]
 X509v3 Subject Key Identifier:
     3C:29:FE:DB:....
 X509v3 Subject Alternative Name:
     DNS:rtr1.rtr, IP Address:203.0.113.56
 X509v3 Router configuration section:
     '{ "prefixes": ["203.0.113.0/24",
                     "198.51.100.0/24",
                     "192.0.2.0/24"]}'
```

Listing 1. Example of a simplified X.509 certificate containing a list of prefixes that a customer AS is allowed to advertise to its provider.

4 Beyond securing the transport of BGP Messages

As discussed in the previous section, securing the BGP transport with protocols such as QUIC and TLS over TCP enables the mitigation of well-known security risks identified for BGP, without the complexity of additional separate solutions reported on Table 1. In addition, the use of TLS can go beyond basic router authentication and data encryption to counter attacks at the transport layer. During the TLS handshake, routers exchange certificates to verify their identities, then establish the BGP session. Although certificates are primarily used to authenticate routers, they can include other fields for various purposes [41]. **We propose to extend X.509 certificates to include router configuration data.** We name these **BGPoST certificates** when they are combined with BGP over a secure transport. Before detailing the configuration added to X.509 certificates, we first remind how eBGP sessions are currently established with a simple example. When a provider AS establishes an eBGP session with a customer AS, the provider's network operator usually configures an import filter [87] ensuring that the customer only announces its own prefixes, or the prefixes of its own customers. Many operators configure these filters manually based on the Internet Routing Registry (IRR) databases. However, adding these filters on a per-eBGP session basis on each router is cumbersome. While tools such as bgpq4 [86] can generate access-lists based on IRR data, the provider's network operator has to push the generated configuration on their routers. Instead, we propose to move this kind of configuration data to the client side using BGPoST certificates. The rest of this section further describes the benefits of using BGPoST certificates between two routers to exchange additional configuration information.

Configuring eBGP Sessions with BGPoST certificates. To enable a BGP session between two routers belonging to different ASes, network operators need to manually configure the peering routers with the AS number of the peer, its IP address, the export and import filters that need to be applied, etc. The configuration of these BGP sessions can be complex and error-prone [62, 69, 90]. We propose to leverage the X.509 certificates exchanged by routers during the BGPoST session establishment. The X.509 certificate format [13] is flexible enough to embed additional information. Furthermore, such certificates [13] are largely used on the web with HTTPS and to secure VPN tunnels, e.g., using OpenVPN [29] or IPSec [35].

We propose a new paradigm where X.509 certificates embed router configuration so that BGP sessions can be automatically configured upon secure transport session establishment. Figure 7

illustrates this new workflow for the simple example mentioned earlier. First, the customer requests a certificate through the "Provider Certificate Generator", which can be a web portal or a REST API. Through this portal, they fill in the prefixes that will be announced. The portal automatically verifies that the prefixes belong to the customer by consulting the Internet Routing Register (IRR), the RPKI or other databases. The portal enables the customer to select a configuration template that has been validated by the ISP [39] but where the customer can specify some parameters like IP addresses, QoS configuration, access lists, etc. This increases the flexibility of the customer network by influencing the configuration of the provider. The customer can still configure its BGP routers as it wishes. The portal generates a certificate that the customer installs on their router. In this simple example, the portal extends the X.509 certificate with a field listing all the prefixes that the customer is authorized to advertise. An example of such a certificate is shown in Listing 1. We chose a JSON representation in our prototype as it is machine-readable and flexible, but other serialization formats could also be used. Finally, the client router initiates a secure transport session towards the provider router. Upon validation of the certificate, the provider router extracts the configuration, e.g., the filter list, and automatically configures its BGP session, e.g., adding the import filter. The same approach can be applied to configure QoS, e.g., prioritize some IP addresses over others, firewall filters and many other tweaks that customers often request from their providers. Operators should agree on YANG [12] models that strictly define the configuration parameters allowed in certificates. Then, routers must validate that the embedded configuration respects the YANG model before applying it.

This solution also allows reconfiguring existing BGP sessions. For example, when a new prefix is assigned to the customer, it can simply request a new certificate from its provider and restart the BGPoST session using this new certificate. Note that thanks to the BGP graceful restart (GR) mechanism [75], it is possible to restart the BGP session and change the TLS certificate without impacting the forwarding of data packets. Of course, a GR implies a reconvergence of the control plane and thus should ideally be performed during maintenance hours. While our current architecture does not modify BGP, the protocol could be further extended to avoid the use of GR by adding a new type of BGP message to exchange the new X.509 certificate without restarting the BGP session. This modification and the evaluation of its impact is left for further works. Operators may request in advance a set of certificates, each covering a different configuration profile. Depending on the state of the network, they can use the one that suits them best.

Securing BGP configuration embedded in X.509 certificates. When a provider sends a certificate to its client, the configuration it contains is exposed as well. A provider may want its internal configuration to stay private. To do so, a provider can encrypt it in two ways.

First, when the configuration is only used to configure a single router, e.g., a router of a stub network, the provider can encrypt the configuration with a symmetric key known only within its AS. Therefore, the client using the certificate will not be able to see the details of the configuration, which is protected in the event of a certificate leak.

Second, when our proposal is used to configure both routers of a peering session, the provider part can still be encrypted with a symmetric key, while the client part of the configuration can be encrypted with a public key provided by the client. In this situation, the client is the only party able to decrypt the configuration stored in the certificate for its router. In both cases, the privacy of the peering policy is enforced by encryption and none of the configuration is readable by third-parties. Note that when customer ASes request their provider to configure their routers, they have to consider the provided configuration. On the other hand, the customer routers are free to ignore the local configuration from the certificate if the operator has not requested such service from its provider.

Provisioning BGPoST certificates for ASes. Using certificates to secure the transport layer of BGP introduces operational concerns regarding their distribution. First, ASes must develop some form of trust to accept certificates issued by external actors, and to leverage them for mutual authentication. Second, failing to provide a certificate can prevent the establishment of BGP sessions. Multiple approaches are possible to address these concerns, but each has drawbacks.

RPKI is the current trust source for BGP as it authenticates Internet resources such as IP prefixes or Autonomous System Provider Authorization (ASPA). On the one hand, this infrastructure is not usable to distribute certificates authenticating BGP routers as it only certifies Internet resources, not identities [15, 57]. On the other hand, RPKI is an orthogonal but complementary security measure to our proposal. Our vision is to secure the transport of BGP messages thanks to TLS as discussed in this paper while RPKI authenticates the Internet objects exchanged in BGP messages.

One could conceive using the current Web PKI to issue and distribute TLS certificates. Services such as Let's Encrypt rely on specific protocols, i.e., ACME, to automatically generate free certificates on demand. ACME proposes three challenge types to validate the identity of the certificate requester by querying (*i*) an HTTP server, (*ii*) a TLS server or (*iii*) a specific DNS record. These challenges aim at validating the requester ownership of the domain name authenticated by the certificate. This implies that BGP routers should be identified with DNS records. One could imagine defining a new kind of DNS records specific for that usage. While this approach could work for plain BGP over TLS, it still presents major drawbacks. First, it assumes that functional routing is established to contact external services and that DNS is deployed and functional. Those are very strong limitations as we are discussing the distribution of TLS certificates to configure BGP sessions, which are themselves required to establish functional routes and access to the DNS. Second, these solutions are extremely centralized, leading to severe security concerns. If one of the main actors of the Web PKI is compromised, BGP sessions will no longer be reliable and global routing could collapse. We argue this goes against the decentralized nature of BGP, putting too much power in the hands of very few actors. While relying on the Web PKI is technically feasible, and could be sufficient for some ASes, we advise operators not to adopt this approach.

Instead, we recommend network operators to develop mutual trust without depending on a third party. To that end, each AS should host its own Certificate Authority, independent of the Web PKI, and form a private PKI with other ASes. Operators already deploy PKIs to handle IPSec certificates [20], mutual TLS (mTLS) for VPNs [34], network access with 802.1x (EAP-TLS [84]), etc. Hence, this should not represent a dramatic investment.

5 Configuring remote BGP services with BGPoST certificates

The use of certificates is not restricted to solely configuring direct BGP sessions as explained in Section 4. By embedding the router configuration in certificates, they also open up new possibilities for configuring secure remote peering sessions, allowing providers to offer new services to their customers. This section presents two use cases for a remote AS to provide services using certificates. The first use case (§5.1) considers a multihomed network that can automatically fall back to another provider link when one of its primary inter-AS links fails. The second use case rethinks the way blackholing is currently performed on the Internet (§5.2). Our solution guarantees the security of the blackhole request through certificates and a secured BGP session.

Due to space constraints, this paper focuses on the remote peering services enabled by embedding configuration data in certificates. We have developed other use cases on top of certificates that are described in Appendix A. It discusses changing the QoS rules of a provider without coordinating with neighboring operators (§A.1), and simplifying the deployment of inter-AS anycast services (§A.2).

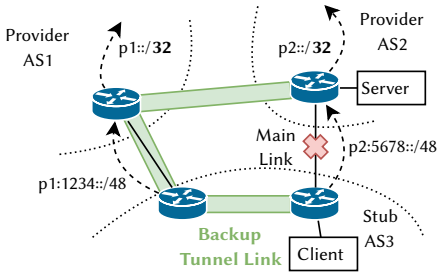


Fig. 8. A multihomed stub can use an interdomain tunnel (Backup Tunnel Link) to protect the Main link with one of its providers.

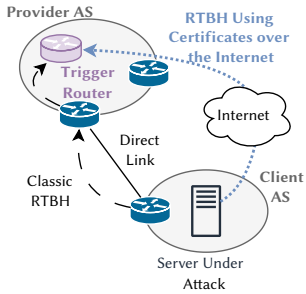


Fig. 9. In classic RTBH, the blackhole request follows the BGP paths to a trigger router. For security reasons, it is only allowed on direct links with the provider. With certificates, the blackhole request can be securely sent through other ASes.

5.1 Resilient IPv6 Multihoming

In 2023, stub networks represented 84% of the ASes connected to the Internet [46]. Stub ASes are often multihomed for redundancy and resiliency. A multihomed stub can use Provider Independent (PI) or Provider Aggregatable (PA) IP prefixes. Most stubs use PI IPv4 prefixes today, following the historical allocation of IP prefixes [43]. With PI prefixes, a multihomed stub advertises its prefixes to its providers. This contributes to the growth of the BGP routing tables, especially when the stub has been allocated IP addresses from different blocks. Furthermore, additional BGP messages must be exchanged when a link fails for BGP to converge.

A better approach from a BGP scalability viewpoint is using Provider Aggregatable IP addresses. PA addresses are rarely used with IPv4, but they are proposed for IPv6 [7]. With PA addresses, a stub connected to two providers, e.g., AS1 and AS2, receives two different IPv6 prefixes. As an example illustrated by Figure 8, a stub can be allocated p1:1234::/48 from AS1 and p2:5678::/48 from AS2. The stub advertises p1:1234::/48 to AS1 and p2:5678::/48 to AS2. However, these two providers do not directly advertise the prefix announced by their customers. Each AS simply advertises its allocated prefix, e.g., p1::/32 for AS1 and p2::/32 for AS2. From a BGP viewpoint, this approach scales much better as only transit ISPs advertise large IPv6 prefixes using BGP. It brings additional benefits in terms of route diversity [23].

However, this approach has one important drawback. When the connection between the stub AS and one of its providers fails, the IP addresses part of the prefix allocated by this provider become unreachable. There are two existing solutions to cope with this problem. In the first solution, one IPv6 address can be allocated per provider to each host inside the stub AS. When one of the providers fails, the other address remains reachable. Multipath transport protocols such as Multipath TCP [32], SCTP [71] and Multipath QUIC [60] can maintain communication. When the two address are reachable, the hosts can use both addresses. Unfortunately, these protocols are not widely deployed and thus cannot be considered as universally usable today. In the second solution, a stub AS, e.g., AS3, can advertise the IPv6 prefix allocated by AS1 to AS2 when its link with AS1 fails. This requires a special configuration of the routing policies of the provider ASes which is difficult to maintain for network operators. In addition, this increases the size of the global routing table.

To offer a resilient multihoming that also works with PA prefixes, we propose a simpler approach that maintains connectivity when a link with a provider fails. The approach does not require any BGP advertisement nor increase the size of the BGP routing tables. It assumes that the IP addresses used on the peering link between a provider and its customer are part of the prefix advertised by

the provider. This is a common practice among operators. For example, the BGP security guidelines concerning IXP subnets recommend using a routable prefix for the IXP LAN [26].

We propose to create an interdomain protection tunnel between the border routers of the stub and the provider. This tunnel is configured to pass through the second provider. This is illustrated in Figure 8 where AS3 is a stub connected to two providers: AS1 and AS2. In the event of a link failure between AS3 and AS2, packets from and to addresses of AS3 allocated by AS2 are re-routed through a pre-configured backup tunnel. Using provider addresses at the end of the tunnel in AS3 ensures that no routing loops will appear when the backup tunnel is used. This requires that the address is globally routable, as well as the address of the tunnel endpoint in AS2, which is not unusual [2]. This tunnel ensures the continuity of the BGP session and the availability of the customer's PA prefix. However, when configured manually, this solution is not practical for operators.

With certificates, these tunnels can be configured automatically. To achieve this, the certificate installed on AS3 and sent to AS2 during the BGP session establishment contains the required information to set up a data plane tunnel, such as a Generic Routing Encapsulation (GRE) tunnel [59]. This certificate is generated by AS2 and contains configuration statements to configure both ends of the tunnel, on the border routers of AS2 and AS3. When their link fails, the routing session detects this, e.g., with BFD, and migrates the control and data plane packets through the tunnel pre-configured by this certificate. All existing flows can continue through this tunnel. In addition, as the BGP session is maintained in the tunnel, no BGP Withdraw nor Update are sent to the Internet.

Proof of concept. To demonstrate and test the feasibility of this approach, we set up a client node in the stub network and a server node in the provider network as depicted in Figure 8. The stub and provider establish a BGPoST session. The exchanged certificate contains instructions for creating the two endpoints of the backup tunnel. It instructs the router of AS2 to create the tunnel using a different interface than the one directly connected to the customer. The certificate also configures the endpoint on the stub router in the same manner. Once the TLS session is established, the BGP session begins, and AS3 starts announcing its prefixes to AS2. Then, the client launches an iPerf3 test towards the server. After five seconds, we simulate a link failure by performing an administrative shutdown on the main link. When the customer router detects a "link down" event, data plane traffic and the BGP session migrate to the backup tunnel. We configured the one-way latency of the backup tunnel to 30 ms, and we repeated the experiment 30 times.

Our results report that the iPerf flow seamlessly migrates from the AS2-AS3 link to the backup tunnel with a median interruption time of approximately 62.19 ms. Considering the 30 ms one-way delay we configured, 32.19 ms are needed for the stub and provider routers to detect the failure based on the reaction times of the Linux kernel and BIRD, and migrate the routes through the tunnel. Other approaches using BFD or a fast reroute technique would reduce the duration of this downtime, but this experiment aimed at demonstrating the configuration flexibility that a certificate can provide for multihomed stubs.

The solution presented in this use case only prevents link failures between the routers of the two ASes. However, it can be extended to protect against router failures by creating additional backup tunnels. This would require different routers in the fallback remote AS. This extended solution is beyond the scope of this paper.

5.2 Improved Blackholing Service

BGPoST certificates can also improve Remote-Triggered Black Hole (RTBH) services [96]. RTBH is a method altering BGP tables to counter Distributed Denial of Service (DDoS) attacks targeting client networks, such that the unwanted traffic is dropped before reaching the target hosts. Some operators or transit providers provide this blackholing service for their customers, as illustrated in Figure 9. When a client AS detects a DDoS attack and wishes to block all packets coming

from a specific provider towards the target address under attack, it informs the provider AS by re-announcing the corresponding route with a special BGP community. This requests the trigger router to insert a special route on the border routers of the provider to drop packets towards the prefix under attack. This solution is widely used, but researchers have shown that it suffers from several problems [68] because such blackholing requests are not authenticated [65]. First, BGP routes can be inadvertently or maliciously manipulated [5, 19]. Second, a third party AS can trigger such remote blackholing for particular prefixes [88]. Using these two vulnerabilities, an attacker could blackhole a given prefix. Also, an experiment we performed on the Internet revealed that some ASes filter BGP communities, making the use of BGP communities not the best choice to implement RTBH across the whole Internet.

Using BGPoST certificates, we propose a new method for providers to support RTBH filtering. Instead of extracting RTBH information from received BGP communities, a provider can provide RTBH service to *any AS* even when they are not directly connected. Using the certificate generated by the provider, the client can authenticate itself to them and request RTBH. Consider a Tier-1 provider that carries significant traffic and wants to support RTBH. It can provide X.509 certificates to customers to later establish a secure remote BGP session. The certificate serves two purposes. First, the certificate authenticates the remote customer to the Tier-1's network, such that its BGP routers only accept BGP sessions using certificates it has issued. Second, the Tier-1 provider includes configuration data in the certificate, such as BGP import filters. They restrict the prefix that the client can send, such that they can be blackholed. When generating the certificate, the Tier-1 provider validates the legitimacy of the client prefix for blackholing. To do so, the Tier-1 ISP can check that the prefix is valid using the Route Origin Authorizations (ROAs) object of the Resource Public Key Infrastructure (RPKI). ROAs are cryptographically signed objects that certify whether a prefix is authorized to be advertised by an AS.

During an attack, the client initiates a BGP session towards the Tier-1 certificate using the provided certificate. Any route announced on this BGP session that are accepted by the import filters in the certificate are then blackholed. The blackhole lasts until the client removes the advertisement from the BGP session or the session ends.

The advantage of this advanced RTBH is that blackholing can be requested closer to the source of the attack and not only to immediate AS peers. Unlike traditional RTBH, the traffic can be selectively dropped in multiple AS simultaneously. In cases where the victim AS is not aware of the origins of a DDoS, it can leverage services such as Cymru [4] that offers global RTBH. Multiple providers have established BGP sessions with Cymru, which acts as a broker when it receives a RTBH request, i.e., it distributes the request to the connected ASes. Our solution may be used to secure the initial RTBH request towards Cymru, then to secure the requests between Cymru and the remote providers.

Proof of concept. To demonstrate our improved RTBH service, we performed some tests on the Internet. We set up two BGP routers, one located in Virginia, United States, and the other connected to our provider in Belgium, which acts as a trigger router.

From our BGP session with the router in Virginia, we advertise an IPv4 prefix with a community that indicates a black hole for the BGP router in Belgium. We choose to create the BGP peering over two different continents so that the BGP messages traverses multiple BGP peers and multiple ASes on the path between the trigger router and the router advertising the black hole route.

We evaluate two techniques. In the first one, named "Classic Blackhole", we use the traditional hop-by-hop BGP session. In this configuration, each router announces the blackhole request to its neighboring BGP routers until the message reaches our BGP router in Belgium. This method is not secure as the BGP message passes through untrusted routers. However, it serves as a baseline to compare with our proposed solution. The second one, "Dynamic Secure Blackhole", presents

our improved RTBH variant. When a DDoS attack is detected, the router in Virginia establishes a secure multi-hop BGP session using our prototype directly with our BGP router in Belgium.

This direct connection ensures that the advertisement remains secure and eliminates the need to propagate the blackhole route through multiple BGP routers. Unlike the traditional RTBH, this secure session avoids the intermediate BGP routers.

For the classical solution, we measure the time needed to propagate the BGP update hop-by-hop over BGP sessions between the two BGP routers. For the dynamic solution, we measure the time taken to establish the remote BGP session (i.e., starting from the first SYN or QUIC initial packet) until the BGP update is received. With classic RTBH, the median time to advertise the route is 1.59 s, whereas, with the dynamic solution, it takes 0.39 s to establish the secure BGP session and announce the black hole. We also observe that the dynamic solution is more stable than the classical one with values varying from 0.39 s to 0.42 s (dynamic) and 0.8 s to 3 s (classical). Indeed, the variance is smaller with fewer BGP routers processing the update in our approach. An outlier we observed for the classical approach had a propagation delay of 30 seconds. These delays can be explained by the use of timers such as MRAI [27, 36, 89] or by a greater influx of routes received by intermediate BGP routers, slowing down the propagation of the BGP update for our prefix.

6 Discussion

In Section 5, we demonstrated two new features enabled by BGPoST certificates. We believe that using QUIC or TLS over TCP as the transport layer of routing protocols can bring additional benefits to distributed routing protocols. Our current prototypes are a first step towards this goal. They open new directions for researchers, network operators, and eventually the IETF. We discuss some of these directions in this section.

Transitioning from BGP over TCP to BGPoST. During the transition to the use of secure transport, there will indeed be a period during which some routers will use TCP and others will be also capable of using TCP/TLS or QUIC. The operator will have the choice of what to do with unsecured BGP sessions. It is akin to the RPKI/ROV problem, in which some prefixes are still not protected by the ROA. The operator can accept or refuse prefixes that are not protected by an ROA. They can also choose to apply a similar policy regarding BGP over TCP. Note that leveraging TLS is sufficient to enable BGPoST, i.e., it only depends on plain X.509 certificates and does not imply embedding router configuration within them.

Semantics for BGP configuration inside X.509 certificates. On the other hand, BGPoST is required to use BGP router autoconfiguration with X.509 certificates, but network operators can use it without autoconfiguration in X.509 certificates. Once the vast majority of routers have adopted BGPoST with traditional X.509 certificates, some operators can offer their customers the use of X.509 certificates to automate the creation of BGP sessions. Our BGPoST prototype uses QUIC or TLS to exchange X.509 certificates which can distribute prefix lists, filters and other types of information in some of their fields. Network operators and the IETF should discuss and agree on semantics defining the information exchanged inside certificates. These certificates could carry different types of information, from prefix lists to traffic engineering requirements. The use of YANG models [12] could serve as a starting point for defining a standard representation of the actions and information that can be included in the certificate.

Optimizing TCP-AO when combined with TLS. Currently, TCP-AO authenticates the entire TCP segment, while TLS only authenticates the TCP byte stream. However, when TLS is used, the TCP header remains the only exposed and vulnerable part to attacks. To improve this, TCP-AO could be modified to solely authenticate the TCP header. This may reduce the CPU cost of authenticating TCP segments.

7 Related Work

The use of QUIC and TLS in BGP. The IETF has started to work on using QUIC to carry BGP messages [79, 80]. Our implementation of BGP over QUIC is partially aligned with this effort. At the time of writing, after discussion with the draft authors and to the best of our knowledge, we are the first to have an implementation of BoQ. The use cases described in this paper go beyond the current IETF discussions. In parallel, another IETF draft has been proposed to secure a BGP session using TLS on top of TCP [100]. Our implementation of BGP over TLS is consistent with this draft.

Securing BGP messages. Our prototypes of BGP over QUIC and TLS secure the transport of a BGP session such that an external attacker cannot inject BGP messages. Another paper also focused on encrypting BGP messages by directly modifying the protocol itself [85]. Their contribution heavily modifies the functioning of BGP and the message that it exchange. Our proposal only require to change the transport of BGP messages. To secure routing information itself, BGP over TLS or QUIC can be combined with other BGP extensions such as BGPsec [58] and RPKI ROA [47]. These extensions provide security against routing changes made by ASes. Several approaches have been proposed to address the lack of security for routing information, including SPV [42], soBGP [98], S-BGP [54], PGBGP [50], and IRV [38].

Secure tunnels for BGP. Securing the transport of routing messages can be also made with IPSec [52, 53, 83] or any other secured tunneling protocol such as Wireguard [24] or OpenVPN [29]. These protocols encapsulate routing messages in an encrypted payload. In the case of IPSec, the Internet Key Exchange (IKE) [51] protocol (*i*) authenticates the two parties with Pre-Shared Keys (PSKs) or X.509 certificates and (*ii*) negotiate the keys used to encrypt the payload. The routers can then securely send their data as the encrypted tunnel provides authenticity, message replay prevention, confidentiality and integrity. QUIC or TLS with TCP-AO provides the same security guarantees but reduces the configuration overhead, as there is no need for an extra protocol to encrypt and authenticate data.

Router autoconfiguration. The IETF also proposed autoconfiguration for BGP sessions [17] using the Layer 3 Neighbor Discovery protocol [16]. This approach is intended for controlled environments, such as data centers. Our approach allows authenticating the configuration such that it can be used in other environments, e.g., remote peering.

8 Conclusion

This paper proposes to replace the BGP transport protocol with a secure transport protocol. Instead of using a simple TCP connection to exchange routing messages, we have modified the BIRD routing daemon to allow BGP to establish QUIC or TCP/TLS (coupled with TCP-AO) sessions with a remote BGP speaker. The secure transport layer makes BGP less vulnerable to packet injection attacks, which is an immediate benefit. In addition, BGP can leverage the extensive improvements and expertise of the TLS stack without the need for dedicated security features on top of BGP.

While improved security is a direct benefit of using TLS, we have also shown that BGP can leverage the X.509 certificates exchanged by TLS to propose new or improved services to ISPs. A certificate can be extended to include router configuration, which is extracted and applied to the router receiving the certificate. The configuration is authenticated along with the certificate, such that the router can trust the source of the configuration. This paper illustrates two uses-cases leveraging these certificates to create backup tunnels for multihomed stubs and to suggest an enhanced version of RTBH filtering by allowing any authenticated AS to request a black hole.

The two proposals of this paper can be deployed separately but are not completely orthogonal. The first one specifies how to secure the BGP transport, and the second one explains how to secure BGP configuration. Our solution is to secure both aspects with a single tool, i.e., TLS, instead of relying on different tools to secure the transport of BGP and securly autoconfiguring routers.

Artifacts

The source code for BGP over TLS and BGP over QUIC is available on Github at the following URLs: https://github.com/IPNetworkingLab/BGPoTLS & https://github.com/IPNetworkingLab/BGPoQUIC. The artifacts related to the use cases presented in this paper are available at: https://github.com/IPNetworkingLab/BGPoST-Artifacts.

Acknowledgments

This work has been partially supported by the Walloon Region (Belgium) as part of the funding of the FRFS-WEL-T strategic axis. We would like to thank Randy Bush for his constructive feedback, which has helped to enhance this paper. We thank Maxime Piraux for his proofreading and revisions, which have significantly improved the form and the content of the paper. We would also like to thank the anonymous reviewers and our shepherd, Italo Cunha, for their valuable comments.

References

[1] Josh Aas, Richard Barnes, Benton Case, Zakir Durumeric, Peter Eckersley, Alan Flores-López, J. Alex Halderman, Jacob Hoffman-Andrews, James Kasten, Eric Rescorla, Seth Schoen, and Brad Warren. 2019. Let's Encrypt: An Automated Certificate Authority to Encrypt the Entire Web. In *Proc. of the 2019 ACM SIGSAC Conf. on Computer and Communications Security* (London, United Kingdom) *(CCS '19)*. ACM, New York, NY, USA, 2473–2487. https://doi.org/10.1145/3319535.3363192

[2] Taha Albakour, Oliver Gasser, and Georgios Smaragdakis. 2023. Pushing Alias Resolution to the Limit. In *Proc. of the 2023 ACM on Internet Meas. Conf.* (Montreal QC, Canada) *(IMC '23)*. ACM, New York, NY, USA, 584–590. https://doi.org/10.1145/3618257.3624840

[3] Christopher Allen and Tim Dierks. 1999. The TLS Protocol Version 1.0. RFC 2246. https://doi.org/10.17487/RFC2246

[4] Radu Anghel, Yury Zhauniarovich, and Carlos Gañán. 2024. Who's got my back? Measuring the adoption of an internet-wide BGP RTBH Service. *Proc. of the ACM on Meas. and Anal. of Computing Systems* 8, 1 (2024), 1–25.

[5] Maria Apostolaki, Aviv Zohar, and Laurent Vanbever. 2017. Hijacking Bitcoin: Routing Attacks on Cryptocurrencies. In *2017 IEEE Symp. on Security and Privacy (SP)*. 375–392. https://doi.org/10.1109/SP.2017.29

[6] Alexander Azimov, Eugene Bogomazov, Randy Bush, Keyur Patel, Job Snijders, and Kotikalapudi Sriram. 2024. *BGP AS_PATH Verification Based on Autonomous System Provider Authorization (ASPA) Objects.* Internet-Draft draft-ietf-sidrops-aspa-verification-17. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-ietf-sidrops-aspa-verification/17/ Work in Progress.

[7] Fred Baker, Chris Bowers, and Jen Linkova. 2019. Enterprise Multihoming using Provider-Assigned IPv6 Addresses without Network Prefix Translation: Requirements and Solutions. RFC 8678. https://doi.org/10.17487/RFC8678

[8] Steven Bellovin. 2007. Key Change Strategies for TCP-MD5. RFC 4808. https://doi.org/10.17487/RFC4808

[9] Ondřej Benkovský. 2024. CLI tool for a high QPS DNS benchmark. https://tantalor93.github.io/dnspyre/.

[10] David Bernstein. 2014. Containers and Cloud: From LXC to Docker to Kubernetes. *IEEE Cloud Computing* 1, 3 (2014), 81–84. https://doi.org/10.1109/MCC.2014.51

[11] Mike Bishop. 2022. HTTP/3. RFC 9114. https://doi.org/10.17487/RFC9114

[12] Martin Björklund. 2016. The YANG 1.1 Data Modeling Language. RFC 7950. https://doi.org/10.17487/RFC7950

[13] Sharon Boeyen, Stefan Santesson, Tim Polk, Russ Housley, Stephen Farrell, and David Cooper. 2008. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280. https://doi.org/10.17487/RFC5280

[14] Matt Brown, Ari Fogel, Daniel Halperin, Victor Heorhiadi, Ratul Mahajan, and Todd Millstein. 2023. Lessons from the evolution of the Batfish configuration analysis tool. In *Proc. of the ACM SIGCOMM 2023 Conf.* 122–135.

[15] Randy Bush and Russ Housley. 2022. The 'I' in RPKI Does Not Stand for Identity. RFC 9255. https://doi.org/10.17487/RFC9255

[16] Randy Bush, Russ Housley, Rob Austein, Susan Hares, and Keyur Patel. 2023. *Layer-3 Neighbor Discovery.* Internet-Draft draft-ymbk-idr-l3nd-06. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-ymbk-idr-l3nd/06/ Work in Progress.

[17] Randy Bush and Keyur Patel. 2023. *L3ND Upper-Layer Protocol Configuration.* Internet-Draft draft-ymbk-idr-l3nd-ulpc-07. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-ymbk-idr-l3nd-ulpc/07/ Work in Progress.

[18] Kevin Butler, Toni R Farley, Patrick McDaniel, and Jennifer Rexford. 2009. A survey of BGP security issues and solutions. *Proc. of the IEEE* 98, 1 (2009), 100–122.

[19] Shinyoung Cho, Romain Fontugne, Kenjiro Cho, Alberto Dainotti, and Phillipa Gill. 2019. BGP hijacking classification. In *2019 Network Traffic Meas. and Anal. Conf. (TMA)*. 25–32. https://doi.org/10.23919/TMA.2019.8784511

[20] Cisco Press. 2005. IPSec Authentication and Authorization Models. https://www.ciscopress.com/articles/article.asp?p=421514&seqNum=4

[21] Cisco System, Inc. 2024. Cisco Secure Firewall ASA Series Command Reference, A-H Commands. https://www.cisco.com/c/en/us/td/docs/security/asa/asa-cli-reference/A-H/asa-command-ref-A-H/crypto-a-to-crypto-ir-commands.html#wp2386520637

[22] CZ.NIC Labs. 2022. The BIRD Internet Routing Daemon Project. https://bird.network.cz/

[23] Cedric De Launois, Bruno Quoitin, and Olivier Bonaventure. 2006. Leveraging network performance with IPv6 multihoming and multiple provider-dependent aggregatable prefixes. *Computer Networks* 50, 8 (2006), 1145–1157.

[24] Jason A Donenfeld. 2017. Wireguard: next generation kernel network tunnel.. In *NDSS*. 1–12.

[25] Madory Doug. 2024. RPKI ROV Deployment Reaches Major Milestone. https://web.archive.org/web/20240606161717/https://manrs.org/2024/05/rpki-rov-deployment-reaches-major-milestone/ Accessed: 2024-06-06.

[26] Jerome Durand, Ivan Pepelnjak, and Gert Döring. 2015. BGP Operations and Security. RFC 7454. https://doi.org/10.17487/RFC7454

[27] Alex Fabrikant, Umar Syed, and Jennifer Rexford. 2011. There's something about MRAI: Timing diversity can exponentially worsen BGP convergence. In *2011 Proc. IEEE INFOCOM*. 2975–2983. https://doi.org/10.1109/INFCOM.2011.5935139

[28] M Fanto, Randall Atkinson, Michael Barnes, Vishwas Manral, Russ White, Tony Li, and Manav Bhatia. 2009. OSPFv2 HMAC-SHA Cryptographic Authentication. RFC 5709. https://doi.org/10.17487/RFC5709

[29] Markus Feilner. 2006. *OpenVPN: Building and Integrating Virtual Private Networks: Learn How to Build Secure VPNs Using This Powerful Open Source Application*. Packt Publishing.

[30] Adrienne Porter Felt, Richard Barnes, April King, Chris Palmer, Chris Bentzel, and Parisa Tabriz. 2017. Measuring HTTPS Adoption on the Web. In *26th USENIX Security Symp. (USENIX Security 17)*. USENIX Association, Vancouver, BC, 1323–1338. https://www.usenix.org/Conf./usenixsecurity17/technical-sessions/presentation/felt

[31] Ari Fogel, Stanley Fung, Luis Pedrosa, Meg Walraed-Sullivan, Ramesh Govindan, Ratul Mahajan, and Todd Millstein. 2015. A general approach to network configuration analysis. In *12th USENIX Symp. on Networked Systems Design and Implementation (NSDI 15)*. 469–483.

[32] Alan Ford, Costin Raiciu, Mark J. Handley, Olivier Bonaventure, and Christoph Paasch. 2020. TCP Extensions for Multipath Operation with Multiple Addresses. RFC 8684. https://doi.org/10.17487/RFC8684

[33] Paul Ford-Hutchinson. 2005. Securing FTP with TLS. RFC 4217. https://doi.org/10.17487/RFC4217

[34] Fortinet, Inc. 2024. Site-to-site IPsec VPN with certificate authentication. https://docs.fortinet.com/document/fortigate/5.6.0/cookbook/530530/site-to-site-ipsec-vpn-with-certificate-authentication

[35] Sheila Frankel and Suresh Krishnan. 2011. IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap. RFC 6071. https://doi.org/10.17487/RFC6071

[36] Alberto García-Martínez and Marcelo Bagnulo. 2019. Measuring BGP Route Propagation Times. *IEEE Communications Letters* 23, 12 (2019), 2432–2436. https://doi.org/10.1109/LCOMM.2019.2945964

[37] Vasileios Giotsas, George Nomikos, Vasileios Kotronis, Pavlos Sermpezis, Petros Gigis, Lefteris Manassakis, Christoph Dietzel, Stavros Konstantaras, and Xenofontas Dimitropoulos. 2021. O Peer, Where Art Thou? Uncovering Remote Peering Interconnections at IXPs. *IEEE/ACM Transactions on Networking* 29, 1 (2021), 1–16. https://doi.org/10.1109/TNET.2020.3025945

[38] Geoffrey Goodell, William Aiello, Timothy Griffin, John Ioannidis, Patrick D McDaniel, and Aviel D Rubin. 2003. Working around BGP: an incremental approach to improving security and accuracy in interdomain routing.. In *NDSS*, Vol. 23. 156.

[39] Joel Gottlieb, Albert Greenberg, Jennifer Rexford, and Jia Wang. 2003. Automated provisioning of BGP customers. *IEEE network* 17, 6 (2003), 44–55.

[40] Andy Heffernan. 1998. Protection of BGP Sessions via the TCP MD5 Signature Option. RFC 2385. https://doi.org/10.17487/RFC2385

[41] Russ Housley, Tim Polk, Dr. Warwick S. Ford, and David Solo. 2002. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 3280. https://doi.org/10.17487/RFC3280

[42] Yih-Chun Hu, Adrian Perrig, and Marvin Sirbu. 2004. SPV: Secure Path Vector Routing for Securing BGP. In *Proc. of the 2004 Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications* (Portland, Oregon, USA) *(SIGCOMM '04)*. ACM, New York, NY, USA, 179–192. https://doi.org/10.1145/1015467.1015488

[43] Kim Hubbard, Dr. Jon Postel, Mark Kosters, Daniel Karrenberg, and David R. Conrad. 1996. Internet Registry IP Allocation Guidelines. RFC 2050. https://doi.org/10.17487/RFC2050

[44] Chritsian Huitema. 2022. Minimal implementation of the QUIC protocol. https://github.com/private-octopus/picoquic.

[45] Christian Huitema, Sara Dickinson, and Allison Mankin. 2022. DNS over Dedicated QUIC Connections. RFC 9250. https://doi.org/10.17487/RFC9250
[46] Geoff Huston. 2024. BGP in 2023 – Have we reached Peak IPv4? https://www.potaroo.net/ispcol/2024-01/bgp2023.html.
[47] Geoff Huston and George G. Michaelson. 2012. Validation of Route Origination Using the Resource Certificate Public Key Infrastructure (PKI) and Route Origin Authorizations (ROAs). RFC 6483. https://doi.org/10.17487/RFC6483
[48] Geoff Huston, Mattia Rossi, and Grenville Armitage. 2010. Securing BGP–A literature survey. *IEEE Communications Surveys & Tutorials* 13, 2 (2010), 199–222.
[49] Jana Iyengar and Martin Thomson. 2021. QUIC: A UDP-Based Multiplexed and Secure Transport. RFC 9000. https://doi.org/10.17487/RFC9000
[50] Josh Karlin, Stephanie Forrest, and Jennifer Rexford. 2006. Pretty Good BGP: Improving BGP by Cautiously Adopting Routes. In *Proc. of the 2006 IEEE Int'l Conf. on Network Protocols*. 290–299. https://doi.org/10.1109/ICNP.2006.320179
[51] Charlie Kaufman. 2005. Internet Key Exchange (IKEv2) Protocol. RFC 4306. https://doi.org/10.17487/RFC4306
[52] Stephen Kent. 2005. IP Authentication Header. RFC 4302. https://doi.org/10.17487/RFC4302
[53] Stephen Kent. 2005. IP Encapsulating Security Payload (ESP). RFC 4303. https://doi.org/10.17487/RFC4303
[54] S. Kent, C. Lynn, and K. Seo. 2000. Secure Border Gateway Protocol (S-BGP). *IEEE Journal on Selected Areas in Communications* 18, 4 (2000), 582–592. https://doi.org/10.1109/49.839934
[55] Stephen T Kent, Charles Lynn, Joanne Mikkelson, and Karen Seo. 2000. Secure Border Gateway Protocol (S-BGP)-Real World Performance and Deployment Issues.. In *NDSS*.
[56] Gregory M. Lebovitz, Manav Bhatia, and Brian Weis. 2013. Keying and Authentication for Routing Protocols (KARP) Overview, Threats, and Requirements. RFC 6862. https://doi.org/10.17487/RFC6862
[57] Matt Lepinski and Stephen Kent. 2012. An Infrastructure to Support Secure Internet Routing. RFC 6480. https://doi.org/10.17487/RFC6480
[58] Matt Lepinski and Kotikalapudi Sriram. 2017. BGPsec Protocol Specification. RFC 8205. https://doi.org/10.17487/RFC8205
[59] Tony Li, Dino Farinacci, Stanley P. Hanks, David Meyer, and Paul S. Traina. 2000. Generic Routing Encapsulation (GRE). RFC 2784. https://doi.org/10.17487/RFC2784
[60] Yanmei Liu, Yunfei Ma, Quentin De Coninck, Olivier Bonaventure, Christian Huitema, and Mirja Kühlewind. 2024. *Multipath Extension for QUIC*. Internet-Draft draft-ietf-quic-multipath-08. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-ietf-quic-multipath/08/ Work in Progress.
[61] Kirk Lougheed and Yakov Rekhter. 1989. Border Gateway Protocol (BGP). RFC 1105. https://doi.org/10.17487/RFC1105
[62] Ratul Mahajan, David Wetherall, and Tom Anderson. 2002. Understanding BGP misconfiguration. *ACM SIGCOMM Computer Communication Review* 32, 4 (2002), 3–16.
[63] Fabricio Mazzola, Augusto Setti, Pedro Marcos, and Marinho Barcellos. 2024. Analyzing Remote Peering Deployment and Its Implications for Internet Routing. *IEEE/ACM Transactions on Networking* (2024), 1–10. https://doi.org/10.1109/TNET.2024.3375898
[64] Microsoft. 2022. MsQuic: Cross-platform, C implementation of the IETF QUIC protocol , exposed to C, C++, C# and Rust. https://github.com/microsoft/msquic.
[65] Loïc Miller and Cristel Pelsser. 2019. A Taxonomy of Attacks Using BGP Blackholing. In *European Symp. on Research in Computer Security*. 107–127.
[66] Giovane C. M. Moura, Marco Davids, Caspar Schutijser, Cristian Hesselman, John Heidemann, and Georgios Smaragdakis. 2024. Deep Dive into NTP Pool's Popularity and Mapping. *Proc. ACM Meas. Anal. Comput. Syst.* 8, 1, Article 15 (02 2024), 30 pages. https://doi.org/10.1145/3639041
[67] Sandra L. Murphy. 2006. BGP Security Vulnerabilities Analysis. RFC 4272. https://doi.org/10.17487/RFC4272
[68] Marcin Nawrocki, Jeremias Blendin, Christoph Dietzel, Thomas C Schmidt, and Matthias Wählisch. 2019. Down the black hole: dismantling operational practices of BGP blackholing at IXPs. In *Proc. of the Internet Meas. Conf.* 435–448.
[69] Eugenio Nerio Nemmi, Francesco Sassi, Massimo La Morgia, Cecilia Testart, Alessandro Mei, and Alberto Dainotti. 2021. The parallel lives of Autonomous Systems: ASN Allocations vs. BGP. In *Proc. of the 21st ACM Internet Meas. Conf.* 593–611.
[70] Chris Newman. 1999. Using TLS with IMAP, POP3 and ACAP. RFC 2595. https://doi.org/10.17487/RFC2595
[71] Yoshifumi Nishida, Preethi Natarajan, Armando L. Caro, Paul D. Amer, and karen Nielsen. 2016. SCTP-PF: A Quick Failover Algorithm for the Stream Control Transmission Protocol. RFC 7829. https://doi.org/10.17487/RFC7829
[72] Arnis Parsovs. 2013. Practical Issues with TLS Client Certificate Authentication. Cryptology ePrint Archive, Paper 2013/538. https://eprint.iacr.org/2013/538 https://eprint.iacr.org/2013/538.
[73] Carlos Pignataro, Pekka Savola, David Meyer, Vijay Gill, and John Heasley. 2007. The Generalized TTL Security Mechanism (GTSM). RFC 5082. https://doi.org/10.17487/RFC5082
[74] Maxime Piraux, Olivier Bonaventure, and Thomas Wirtgen. 2024. *Opportunistic TCP-AO with TLS*. Internet-Draft draft-piraux-tcp-ao-tls-01. IETF. https://datatracker.ietf.org/doc/draft-piraux-tcp-ao-tls/01/ Work in Progress.

[75] Yakov Rekhter, John Scudder, Srihari R. Sangli, Enke Chen, and Rex Fernando. 2007. Graceful Restart Mechanism for BGP. RFC 4724. https://doi.org/10.17487/RFC4724

[76] Eric Rescorla. 2000. HTTP Over TLS. RFC 2818. https://doi.org/10.17487/RFC2818

[77] Eric Rescorla. 2018. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446. https://doi.org/10.17487/RFC8446

[78] Eric Rescorla and Tim Dierks. 2008. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246. https://doi.org/10.17487/RFC5246

[79] Alvaro Retana, Yingzhen Qu, Jeffrey Haas, Shuanglong Chen, and Jeff Tantsura. 2023. *BGP over QUIC*. Internet-Draft draft-retana-idr-bgp-quic-01. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-retana-idr-bgp-quic/01/ Work in Progress.

[80] Alvaro Retana, Yingzhen Qu, and Jeff Tantsura. 2022. *Use of Streams in BGP over QUIC*. Internet-Draft draft-retana-idr-bgp-quic-stream-02. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-retana-idr-bgp-quic-stream/02/ Work in Progress.

[81] Routing Information Service (RIS). 2023. rrc01 snapshot. https://data.ris.ripe.net/rrc01/2023.11/updates.20231115.1115.gz

[82] Nicolas Rybowski, Cristel Pelsser, and Olivier Bonaventure. 2024. OFIQUIC: Leveraging QUIC in OSPF for Seamless Network Topology Changes. In *2024 IFIP Networking Conf. (IFIP Networking)*. 368–376. https://doi.org/10.23919/IFIPNetworking62109.2024.10619718

[83] Karen Seo and Stephen Kent. 2005. Security Architecture for the Internet Protocol. RFC 4301. https://doi.org/10.17487/RFC4301

[84] Daniel Simon, Ryan Hurst, and Dr. Bernard D. Aboba. 2008. The EAP-TLS Authentication Protocol. RFC 5216. https://doi.org/10.17487/RFC5216

[85] B.R. Smith and J.J. Garcia-Luna-Aceves. 1996. Securing the border gateway routing protocol. In *Proc. of GLOBECOM'96. 1996 IEEE Global Telecommunications Conf.*, Vol. MiniConfInternet. 81–85. https://doi.org/10.1109/GLOCOM.1996.586129

[86] Job Snijders. 2019. BGP Filter Generator. https://github.com/bgp/bgpq4.

[87] Richard Steenbergen. 2006. IRR Power Tools - A utility for managing Internet Routing Registry (IRR) filters. (2006). Presented at NANOG36 https://archive.nanog.org/meetings/nanog36/presentations/steenbergen.pdf.

[88] Florian Streibelt, Franziska Lichtblau, Robert Beverly, Anja Feldmann, Cristel Pelsser, Georgios Smaragdakis, and Randy Bush. 2018. BGP communities: Even more worms in the routing can. In *Proc. of the Internet Meas. Conf. 2018* (Boston, MA, USA) *(IMC '18)*. ACM, New York, NY, USA, 279–292. https://doi.org/10.1145/3278532.3278557

[89] Renata Teixeira, Steve Uhlig, and Christophe Diot. 2007. BGP Route Propagation between Neighboring Domains. In *Proc. of the 8th Int'l Conf. on Passive and Active Network Meas.* (Louvain-la-Neuve, Belgium) *(PAM'07)*. Springer-Verlag, Berlin, Heidelberg, 11–21.

[90] Cecilia Testart, Philipp Richter, Alistair King, Alberto Dainotti, and David Clark. 2019. Profiling BGP serial hijackers: capturing persistent misbehavior in the global routing table. In *Proc. of the Internet Meas. Conf.* 420–434.

[91] The H2O Project. 2016. TLS 1.3 implementation in C. https://github.com/h2o/picotls.

[92] The kernel development community. 2023. TCP Authentication Option Linux implementation (RFC5925). https://docs.kernel.org/networking/tcp%5Fao.html

[93] The kernel development community. 2024. The kernel's command-line parameters. https://docs.kernel.org/admin-guide/kernel-parameters.html

[94] Martin Thomson and Sean Turner. 2021. Using TLS to Secure QUIC. RFC 9001. https://doi.org/10.17487/RFC9001

[95] Dr. Joseph D. Touch, Ron Bonica, and Allison J. Mankin. 2010. The TCP Authentication Option. RFC 5925. https://doi.org/10.17487/RFC5925

[96] Doughan Turk. 2004. Configuring BGP to Block Denial-of-Service Attacks. RFC 3882. https://doi.org/10.17487/RFC3882

[97] Xiaoyun Wang and Hongbo Yu. 2005. How to Break MD5 and Other Hash Functions. In *Advances in Cryptology – EUROCRYPT 2005*, Ronald Cramer (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 19–35.

[98] Russ White. 2003. *Deployment Considerations for Secure Origin BGP (soBGP)*. Internet-Draft draft-white-sobgp-bgp-deployment-01. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-white-sobgp-bgp-deployment/01/ Work in Progress.

[99] Thomas Wirtgen. 2024. Survey on the configuration of BGP routers. http://hdl.handle.net/2078.1/292356

[100] Thomas Wirtgen and Olivier Bonaventure. 2024. *BGP over TLS/TCP*. Internet-Draft draft-wirtgen-bgp-tls-01. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-wirtgen-bgp-tls/01/ Work in Progress.

[101] Tatu Ylonen. 1995. *The SSH (Secure Shell) Remote Login Protocol*. Internet-Draft draft-ylonen-ssh-protocol-00. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-ylonen-ssh-protocol/00/ Work in Progress.

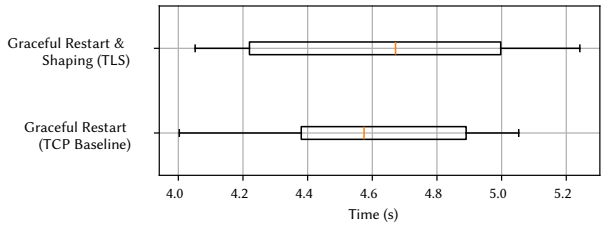


Fig. 10. Time needed to perform a Graceful Restart of the BGP session with a new certificate. The TCP baseline correspond to the time required to only restart the BGP session without any additional configuration.

A Additional use-cases

Since X.509 certificates can contain arbitrary configuration data, the network operator can define non-trivial configurations that would otherwise require manual configuration of the router. We present two additional use cases that are simplified using certificates.

A.1 Dynamic QoS Configuration

Providers offer routing services, but could also provide advanced services such as prioritization of certain traffic classes (e.g., VoIP) or bandwidth negotiation. To provide dynamic Quality of Service (QoS) according to the needs of the customer, coordination between the network operators of the customer and the provider is currently needed. Once the requirements have been agreed on, the operator applies new configurations to the routers either manually or via their management platform. When the customer wishes to change its requirements, the ISP needs to agree and apply the modification, making this process potentially poorly reactive.

With certificates, we provide a mean to automatically carry out configuration changes. Consider the case of a customer wishing to establish a dedicated connection with some cloud provider. As the peering costs vary according to the negotiated bandwidth[1], the customer may be resistant to buy this service at all time but could adapt to different bandwidths scaling with its day and night traffic. With our approach, the customer can request several certificates from the provider, each corresponding to a specific bandwidth limit. When the customer establishes the routing session with the provider during the day, it uses the certificate providing the maximum bandwidth. During the night, the routing session is restarted (for example, by performing a graceful restart) to use the certificate with the reduced bandwidth configuration.

From the operator's point of view, the use of certificates reduces the burden of configuration management and facilitates appropriate billing. In particular, each time a certificate is issued to one router of the provider, our prototype extracts the QoS configuration from the certificate, and logs its use in the system syslog. This logging process enables the implementation of a dynamic billing system that correlates usage with the specific certificate used.

Proof of concept. To demonstrate the feasibility of our approach, we set up a small topology consisting of 4 nodes: a client, a server and two routers. One router plays the role of customer, while the other plays the role of provider. We issued two certificates, the first limits the bandwidth between the two routers to 100 Mbps, while the second limits it to 50 Mbps. After establishing the BGP session, we start an iPerf3 test in UDP mode to saturate the bandwidth. After ten seconds, we perform a graceful restart on the client router using the 50 Mbps certificate. Figure 10 compares the time taken to perform a graceful restart when the new 50 Mbps certificate is applied using TLS to a graceful restart of a regular BGP session over TCP, serving as a baseline. We ran the test 30

[1]See example of pricing for AWS peering in https://aws.amazon.com/directconnect/pricing/?nc1=h_ls

times, measuring the time taken to stop the BGP daemon, restart BGP in graceful mode, apply the shaping rules and complete the graceful restart mechanism.

The regular graceful restart over TCP takes a median time of 4.58 seconds, while a graceful restart on TLS that replaces the certificate takes a median time of 4.67 seconds. We argue this represents a reasonable overhead, especially when considering the time and configuration effort required when performing this operation manually. Our experience also shows that it is possible to dynamically change the QoS configuration by replacing certificates. Bandwidth reduction is a simple example of what can be done with a certificate. In fact, more complex changes to the QoS configuration can be negotiated between BGP peers using certificates. In addition, the installation of QoS rules can be verified using a network verification tool such as batfish [14, 31].

A.2 Flexible Scaling of Anycast Services

To dynamically adapt to the load that a service may experience over time, service administrators can use methods that scale a service across multiple replicas. For example, the container orchestrator Kubernetes [10] can be configured to increase or decrease the number of "pods" (i.e., units of computation) based on the service load. Kubernetes works well when the administrator controls the Kubernetes stack and the servers on which it can be deployed. However, when the service is distributed across multiple ASes, the coordination to deploy Kubernetes can be cumbersome. Other systems, such as the NTP pool project tackles Internet-scale deployment by asking volunteers to host an NTP server [66], then to publish its IP address and location via NTP Pool web interface. The NTP Pool updates the DNS to geographically redirect users to the nearest NTP server.

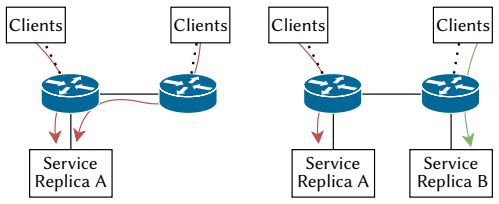
Another method of scaling an inter-AS service is to use anycast addresses. An anycast service distributes its load across multiple servers using the same IP address. Using an anycast address also allows users to be routed to the nearest server based on their location. A typical example of an anycast service is the provision of DNS servers, such that users can access the nearest DNS server. Integrating new servers into an anycast service can require a lot of coordination between networks. For example, while Netflix caches are not directly related to anycast, adding them into an ISP network require similar actions to adding a DNS root server[2]. The network operator is responsible for manually configuring their equipment and the BGP session with the appliance. They must also define the routing policies and provide all the necessary information to establish the session and enable the appliance to be used for the service.

From these examples it can be seen that there are several techniques available for configuring a dynamically scaled service. Unfortunately, there is no generic approach to configuring them, as custom policies can be applied per replicated service.

We propose a method using BGP and certificates achieving a consistent way of scaling inter-AS services. To add a service from a service provider to a given network, the operator requests a certificate containing parameters for the required routing session to the service provider. The operator installs the required software for the service on a given server and starts a BGP daemon on this server. It establishes a BGP session towards the service provider with this certificate. When it is established, the service provider can send the anycast address that should be advertised by the network, such that the server can accept clients connections. A monitoring system on the server is included to ensure that the service is running correctly, such that the anycast route can be announced or withdrawn accordingly.

Proof of concept. To demonstrate the feasibility of the approach, we have created a small experimental setup as illustrated by Figure 11. We deploy two replica nodes, each connected to one router. Each node contains the NSD authoritative DNS server (v4.6.1), which serves a single `A`

[2]See https://openconnect.zendesk.com/hc/en-us/articles/360035533071-Network-configuration



(a) One Replica. (b) Two Replicas.

Fig. 11. With certificates, Replica B can be dynamically added to spread the load of a service.

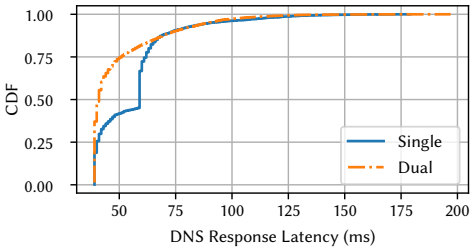


Fig. 12. When adding another replica using certificates, the latency of the DNS service to users decreases.

record. We then connect two clients nodes, also connected each to the routers. These two nodes simulate multiple clients using the `dnspyre` tool [9]. In the first scenario (Figure 11a) "Single", all clients are routed to the only available replica on the left router. In the second scenario (Figure 11b) "Dual", clients coming from the right router are seamlessly redirected to the Replica B. The load is adapted for each scenario, such that each DNS replica receives 2000 concurrent requests over 13 seconds in all scenarios. We configured each links of the topology with a 10 ms delay.

Figure 12 shows the response time observed in both cases. The blue curve, labelled "Single", contains two notable sub-curves, which matches the time taken to reach the DNS servers. The first sub-curve, with a median of 41 ms, corresponds to queries served for the dnspyre client on the left, while the second sub-curve, with a median of 61 ms, corresponds to the dnspyre client on the right. The average response time for the "Single" experiment is 57.5 ms, compared with 50.5 ms when the DNS service is replicated on the right. The orange curve shows that customers are served more quickly when reaching the nearest DNS server. The use of certificates allows the service to adapt dynamically to the state of the network and the location where the service is most in demand to achieve a better service.

Received June 2024; revised September 2024; accepted October 2024