View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Aaltodoc Publication Archive

Aalto University
School of Science
Master's Programme in Security and Mobile Computing

Dmitri Tsumak

Securing BGP using blockchain technology

Master's Thesis
Espoo, November 12, 2018

Supervisors: Professor Tuomas Aura, Aalto University
Professor Panagiotis Papadimitratos, KTH

Aalto University
School of Science
Master's Programme in Security and Mobile Computing

ABSTRACT OF
MASTER'S THESIS

Author:	Dmitri Tsumak		
Title:	Securing BGP using blockchain technology		
Date:	November 12, 2018	**Pages:**	61
Major:	Security and Mobile Computing		
Supervisors:	Professor Tuomas Aura Professor Panagiotis Papadimitratos		

The Border Gateway Protocol (BGP) is an important routing protocol used to exchange routing information among autonomous systems on the Internet. The BGP version 4 does not include specific protection mechanisms against attacks or deliberate errors that could cause disruptions of routing behavior. There were several securing solutions developed to mitigate security issues of BGP. In this thesis, current secure solutions are reviewed and evaluated against a list of security and deployment requirements. Furthermore, a new BGP securing solution is proposed which uses blockchain technology and smart contracts to exchange information required for messages validation among peers. This allows to decouple security-related data from the protocol itself and fix the problems introduced in other BGP solutions.

Keywords:	border gateway protocol, security, blockchain, smart contracts, networking
Language:	English

Contents

List of Figures

Listings

Chapter 1

Introduction

The systems of IP networks managed by one or more Internet service providers are called autonomous systems. They have common routing policies with the Internet and usually consist of routers and switches interconnected with each other. There are two types of protocols used in autonomous systems: IGP and EGP.

Interior Gateway Protocols (IGP) are used to redistribute routing information within an autonomous system. Examples of such protocols are RIP, EIGRP, OSPF, IS-IS. They distribute routes based on link quality, such as bandwidth and latency, or are used for load balancing. They make routers aware of routes to different subnets in a network.

By contrast, exterior gateway protocols (EGP) are used to redistribute routing information between autonomous systems. The only EGP protocol used nowadays is a Border Gateway Protocol (BGP). BGP v4 is currently the latest version and is a standard protocol for Internet routing. It is used by all of the Internet service providers (ISPs) to exchange routing information [11]. BGP is referred to interior BGP (iBGP) when the routes are exchanged within an AS and exterior BGP (eBGP) when they are exchanged between different ASes. This research focuses on the concepts of eBGP.

BGP was not designed with security in mind. As a result, it is vulnerable to multiple attacks and is not able to deal with errors coming from outside that could cause disruptions of routing behavior. Most of such attacks are based on the fact that there is no way in BGP to validate the messages coming from its peers and they are considered valid by default. This issue still persists and there were many hijacking accidents recorded which have caused a large amount of damage. There were several solutions developed to mitigate the security issues of BGP. However, they address the issues in different ways and have some performance, deployability and security tradeoffs. In spite of the large variety of different secure solutions, the BGP attacks have still occurred. For example, the last attack on Iran Telecommunication Company (AS58224) hijacked 10 prefixes of Telegram Messenger. As a result, the existing solutions do not provide full protection against hijacking attacks and there is still some space for work to be done.

In this thesis, a new design is proposed for fixing problems introduced in BGP and its

modifications. The solution uses blockchain and smart contracts to exchange information required by BGP protocol to validate the messages between autonomous systems. It aims to fix the issues introduced in other secure BGP solutions by decoupling the data required for BGP UPDATE messages verification from protocol to smart contracts. This allows ASes to share the data instantly and other ASes can upon synchronizing the changes be confident in the validity and integrity of the data stored in a blockchain. In combination with Quorum blockchain platform and the topology similar to IRV protocol, the deployment of such approach does not require much effort and the performance is very promising. The flexibility of smart contracts allows ASes to share any kind of information with other ASes in a network.

The thesis consists of six chapters. The first chapter is the current one. The second chapter gives a background about the BGP protocol, summarizes the security, deployability requirements. The third chapter reviews existing BGP secure solutions and evaluates them against the requirements. The fourth chapter gives a detailed description of a developed solution. The fifth chapter evaluates the security, deployment and performance properties of a new solution. The last chapter gives the summary of the conducted research and gives an overview of the future work.

Chapter 2

Background

This chapter gives an overview of Border Gateway Protocol and lists the requirements for its modified versions.

2.1 Border Gateway Protocol

Border Gateway Protocol (BGP) is a routing protocol used to exchange network routing and reachability information among routers between different ASes. It is classified as a dynamic protocol which means it can route the traffic based on a network conditions or traffic properties. It does not require knowledge of the whole network topology and, as a result, is classified as an exterior gateway protocol (EGP). It is a critical requirement to make inter-domain routing scalable and efficient. The BGP is also called a path vector protocol (PVP) which means that it can manage dynamic paths information, detect and prevent loops in routing paths. The protocol which is classified as EGP and PVP has the following characteristics:

- flexibility of router filtration system
- protection against routing loops
- management of routing priorities
- high stability

The initial version of BGP was proposed in 1989 [10] and was improved up to the fourth version released in 2006 [12]. BGP is also largely used in scalable private IP networks and for connecting hosts to multiple networks for a better network redundancy.

IP prefixes and AS numbers

Each AS has its own 16-bit or 32-bit unique identification number called ASN. These numbers can be received from Regional Internet Registry (RIR) or Local Internet Registry (LIR), which are globally managed by the Internet Assigned Numbers Authority (IANA). It is a nonprofit organization which controls global IP address allocation, root zones in DNS, and other IP-related values.

The public AS numbers are in the range of 1 to 64511 and are advertised by ASes on the Internet [14]. The private AS numbers are in the range of 64512 to 65535 and are used by public ASes for identifying underlying managed ASes. The ASN of the private AS is replaced with its parent public ASN during the path advertisement to other public ASes [14]. In addition, IANA and its delegated registries are responsible for assigning and managing IP prefixes for ASes. For example, if ISP wants to get a new IP prefix, it should apply for it to IANA authorities. These prefixes can be split into smaller sizes and assigned to ISP's managed networks.

Protocol concept

BGP forwards and routes the data packets based on policies defined by network administrators. The policies are applied based on packet size, list of connected ASes or any other criteria. The exchanged routing information is used to construct paths to all other ASes on the Internet. It fills the routing table with gateways to other networks by receiving paths from the neighbor ASes. If there are multiple paths to the same network, it chooses the best based on some criteria like distance in hops, network policies, or rule-sets configured by a network administrator. The path is a chain of ASes which have to be traversed to reach the desired network. The best paths, located in a routing table, are appended with current AS number and advertised to peers, so they could know how to reach other networks on the Internet. The routers which advertise the path to their AS are called BGP speakers.

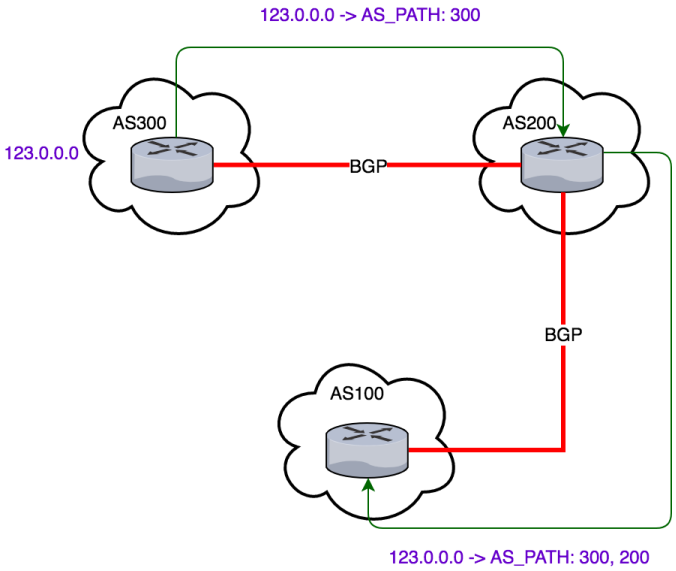

Figure 2.1: Autonomoys system path advertisement

In Figure 2.1 the AS300 is advertising its prefix 123.0.0.0/24:

1. AS300 advertises its prefix 123.0.0.0/24 to the peer called AS200 and sends its AS number as a final destination.
2. AS200 checks that it is the best path it knows to this network, adds path's gateway (received in the same message as the path) to the routing table.
3. AS200 advertises the new path to its peer AS100 and appends its AS number.
4. AS100 adds the gateway to 123.0.0.0/24 (it will point to AS200) to its routing table and is able to send packets to the network 123.0.0.0/24.

The connection between peers is established in the following way :

- BGP peers start working in IDLE mode.
- BGP speakers listen to incoming connections and initiate connections to its peer. This state is called CONNECT. BGP uses TCP protocol to make connections reliable.
- BGP changes state to ACTIVE when the router finds it peers. In ACTIVE state peers try to establish a TCP connection.

- After the connection is established routers change the BGP state to OPEN and start to negotiate BGP parameters, such as protocol version, Hold and Keepalive timers. In this state, they also check whether the ASN of the peer is the same as preconfigured on a router and whether its router ID is different.
- In case router encounters some error, it sends NOTIFICATION message and returns to the IDLE state. If there are no issues in negotiating the parameters, routers change their state to ESTABLISHED and start to exchange UPDATE messages. KEEPALIVE messages are used to confirm connectivity between peers.
- ROUTE REFRESH messages are used to query peers for all the routers without restarting of BGP protocol.

When the connection is established, peers start to exchange UPDATE messages. These messages contain the following routing information:

- AS_PATH - shows AS numbers in the route.
- NEXT_HOP - the gateway for this path.
- ORIGIN - shows the nature of the route.
- Network Layer Reachability Information - contains information about what networks can be reached with this path. For example, 123.0.0.0/24.

Prefix and ASN hijacking

BGP has no mechanism to check the correctness of UPDATE messages sent by the peers. Namely, peers can advertise any prefix and any AS number and they will not be checked at the receiving AS. The dishonest peer could also advertise a more specific prefix than the one advertised by an honest AS.

ASes receiving invalid announcement may select the invalid routes and direct traffic towards malicious AS. As a result, packets will not reach the destination or can be dropped which could lead to the black hole.

Path hijacking

The attacker announces incorrect paths to other ASes. As a result, attacker's peers will send packets to this malicious AS if its announced path is shorter or more preferred.

BGP lacks integrity, confidentiality, and timeliness. BGP does not have proper route selection mechanism, which makes it difficult to detect and reject bad routes. This makes it vulnerable to various Man in the middle and Denial of Service attacks [14]:

- Attacker can inject, modify BGP packets exchanged between ASes. This could lead to the invalid routing table, connection dropouts or DoS.
- Attacker can disable an AS by using a DDoS attack on TCP. As a result, AS will not be able to accept any incoming connection and will force other ASes to traverse their traffic through different paths.
- Attacker can influence path selection by changing MED term or ORIGIN term value in UPDATE messages. Victim's network policies can take into account these attributes and prefer different paths.
- Attacker can perform replay attacks by storing the UPDATE messages sent by ASes and resubmit them again to modify paths based on his preferences.
- Attacker can use one of the attacks above as part of another attack.

Summary

BGP is a fundamental protocol which makes routing between ASes on the Internet possible. However, it was not designed with security in mind and requires network administrators to use other tools and/or protocols to regulate that.

2.2 Requirements

This section lists security and deployment requirements for BGP modified solutions [7].

Security

S-1 The BGP announcement receiver must be able to determine that the first AS in the received path was authorised to announce the prefix. Namely, it should be able to check that there is connection between the first AS number and IP prefix in a path.

S-2 Replay of BGP UPDATE messages should not be possible. However, there could be some mechanism for setting up a window during which the message is still valid.

S-3 The secure version of BGP UPDATE messages should provide up to date information about paths between ASes.

S-4 The secure version of BGP UPDATE messages should provide up to date information about prefix allocations.

S-5 The secure version of BGP should be able to verify whether applied path is still valid.

S-6 The secure version of BGP should be able to verify whether applied path prefix still belongs to the same AS.

S-7 The secure version of BGP should provide link layer integrity between peers. Namely, packets injection, deletion, modification or replay should be prevented.

S-8 The secure version of BGP should reveal information to peers only required to ascertain the correctness of messages. As a result, it should reveal peering, customer/provider relationships as less as possible.

S-9 The secure version of BGP should signal or emit logs about security exceptions which are important for network operators.

S-10 The storage of routing information database should be secured by authentication and imply periodic reauthentication.

S-11 The secure version of BGP should use secure crypthographic algorithms.

S-12 The secure version of BGP should be resistant to downgrade attacks.

Deployment

D-1 The secure version of BGP must be backward compatible with BGP and other versions of secure BGP so that it could be deployed incrementally.

D-2 The secure version of BGP must be backward compatible with the way messages are formatted, processed and transmitted, so they could be parsed in environments with mixed BGP versions.

D-3 The secure version of BGP should not possess large memory, storage and CPU overhead on routers.

D-4 The secure version of BGP should allow peers to configure use of security mechanism on per-peer basis.

D-5 The secure version of BGP should allow peers to apply their custom routing policies on received paths to determine the preferred one.

D-6 The secure version of BGP should be easy to deploy, maintain and remove.

Chapter 3

Related Work

In the following sections, a survey of several BGP modified solutions will be given. Each solution will be evaluated against the requirements defined in the previous chapter and the missing ones will be marked with ✗.

3.1 Secure Border Gateway Protocol

Secure Border Gateway Protocol (S-BGP) is a modification of the BGP protocol targeted to fix its security issues. S-BGP uses digital signatures and X.509 certificates to generate and validate BGP UPDATE messages advertised by ASes [14]. Its security mechanisms prevent a malicious AS from hijacking invalid paths and routing data to other ASes on the Internet [20].

S-BGP consists of three major security elements:

- Public key infrastructure (PKI)
- Attestations (Address and Route)
- Internet Protocol Security (IPSec)

Each of these elements will be reviewed in the following sections.

Public key infrastructure

S-BGP uses public key infrastructure (PKI) and digital signatures to authorize ASes, their BGP speakers, IP prefixes and their AS numbers. S-BGP uses PKI and digital signatures to establish the authenticity of the binding between a public key and its owner.

Each entity in S-BGP has a private, public key pair. IANA is the root certificate authority which has a self-signed certificate and is trusted by all other entities. IANA

allocates AS numbers, address blocks to Regional Internet Registries (RIRs), includes them in the certificates requests and signs with its private key. RIRs, in turn, perform the same action for Local Internet Registries (LIRs) and sign their certificate requests by their private keys. RIRs are responsible for issuing certificates for ASes, which in turn can allocate certificates for their routers as their representatives [20]. As a result, S-BGP certificates can be represented in form of a tree, where the root is IANA and leaves are routers of ASes.

Each S-BGP certificate includes at least the following information:

- AS number of certificate owner.
- IP blocks assigned to it by the certificate issuer.
- Public key of certificate owner.
- Digital signature of the certificate issuer.

AS receives a signed UPDATE message from its peer and can verify the authenticity of the message by validating AS's certificate and checking whether the message signature corresponds to the public key in the certificate.

Attestation

Attestations are used to verify the authenticity and integrity of the data. They encapsulate digital signatures within UPDATE messages [19]. The integrity of the message is achieved by checking the digital signature in the UPDATE message and verifying whether it corresponds to sending AS certificate. Attestations are also used to verify the path and IP prefixes authenticity.

Path attestation

Path attestations are used to verify the authenticity of the advertised routes. Each AS signs paths it sends to its peers, so they could verify that AS was really authorized to advertise them. In order to advertise the path further, receiving AS adds its AS number to the path and signs previously attached signature. The receiving AS validates the route by checking the signature and verifying that it corresponds to the certificate of source AS. Signatures are checked recursively for each AS in path. As a result, the following information is verified:

- There were no ASes added or removed from/to the path.
- The order of ASes in a path was not modified.

Path attestations are included in UPDATE message as an optional attribute to make ASes which do not speak S-BGP backward compatible with their peers.

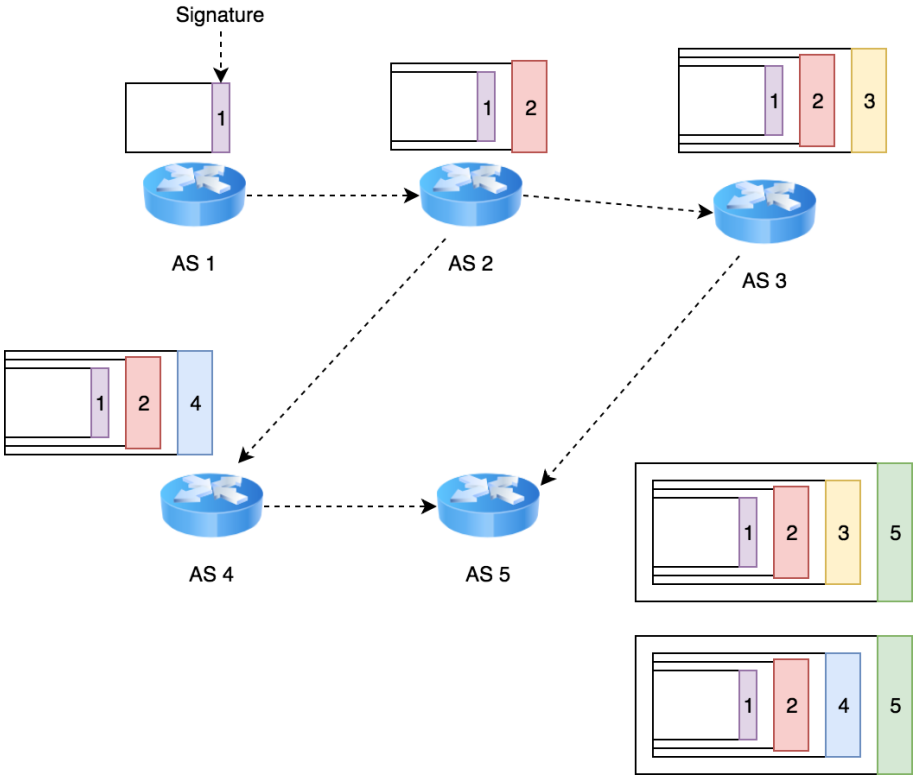

Figure 3.1: Path attestation in S-BGP (based on the Figure 5 from [14])

In figure 3.1 UPDATE messages are passed between peers. The peers validate messages and sign them before sending to other neighbors.

Prefix attestation

Prefix attestations verify whether the advertised prefix in an UPDATE message actually belongs to the AS. The attestation is a digitally signed statement which is distributed using BGP or any other way [14]. They prove the delegation of the prefix from one AS to another. The validation is done by checking the signatures chain from IANA to the advertising AS.

Internet Protocol Security

Internet Protocol Security (IPSec) is a protocol used to secure routing over Internet Protocol (IP) networks. It authenticates and encrypts packets sent over a network using PKI.

The cryptographic keys are negotiated during session establishment and used until the end of it. IPsec can be used to secure connection between hosts, gateways or hosts and gateways.

IPSec provides point-to-point security for traffic between BGP routers. It prevents DoS, replay attacks, ensures integrity and confidentiality of messages on the IP level.

Problems

S-BGP is one of the best BGP securing solution. It eliminates most of the security problems in the BGP protocol and provides better security for exchanging the messages between ASes. However, it still lacks timeliness which means it is exposed to replay attacks on UPDATE messages (requirement **S-2**). Namely, malicious AS can resend an old UPDATE message which was withdrawn before or it can suppress UPDATE messages for its peers, so they will not be able to get the correct paths (requirements **S-5,6**). In addition, the malicious peer can remove signatures from the messages and further ASes will not be able to verify them (requirement **S-12**).

One of the biggest issues with S-BGP is its complexity. It is expensive to adopt this protocol and the performance is reduced due to attestations and signatures' calculation for each UPDATE message (requirement **D-3**). According to survey [14], ASes exchange paths two times slower with S-BGP compared to BGP. However, it is possible to reduce the time required for path convergence by validating only paths which are selected as preferred. Also, S-BGP has some storage requirements for route attestations which makes its usage more complex (requirement **D-6**).

✗ Security: S-2, S-5, S-6, S-12

✗ Deployment: D-3, D-6

Summary

S-BGP provides the most comprehensive guarantees for secure BGP communications [14]. The exchanged paths and IP prefixes are validated and proved to be valid at destination ASes by using attestations. Also, update integrity and confidentiality is achieved by using the IPSec protocol. However, high S-BGP security has some tradeoffs. Namely, its storage requirements for attestations and reduced performance due to signatures validation on every UPDATE message can discourage ISPs from deploying it on their ASes.

3.2 Secure Origin Border Gateway Protocol

Secure Origin Border Gateway Protocol (soBGP) is a BGP modification intended to fix BGP security problems and make communication between peers more reliable. Similarly to S-BGP, it uses PKI to authenticate and authorize entities in the network [14]. The solution was proposed by Cisco Systems in 2002 and intended to validate and authorize the data carried within BGP and protect against misconfiguration or malicious insertion of invalid data into the Internet routing system [21].

soBGP uses four types of certificates to verify prefixes and paths of peers:

- Entity Certificate
- Authorization Certificate
- Prefix Policy Certificate
- AS Policy Certificate

The purpose of these certificates will be described in the following sections.

Entity Certificate

Entity Certificate (EntityCert) is an X.509v3 format certificate which binds AS public key to its ASN. The private key is stored in ASes and used to sign certificates [21] delegated to child entities such as routers and customers. AS has to store the private key in a most secure way (probably offline) to ensure nobody has an access to it. If the private key is compromised, the attacker can issue trusted certificates for malicious entities in a network which will be trusted by other peers.

EntityCert is signed by a third party which validates whether the key pair actually belongs to an ASes. Similarly to S-BGP, there are several certificate authorities managed by IANA which are trusted by default and their certificates are not validated by network entities. As a result, soBGP forms a web of trust (see figure 3.2) which starts with a top-level service provider [21] and ends with AS border routers.

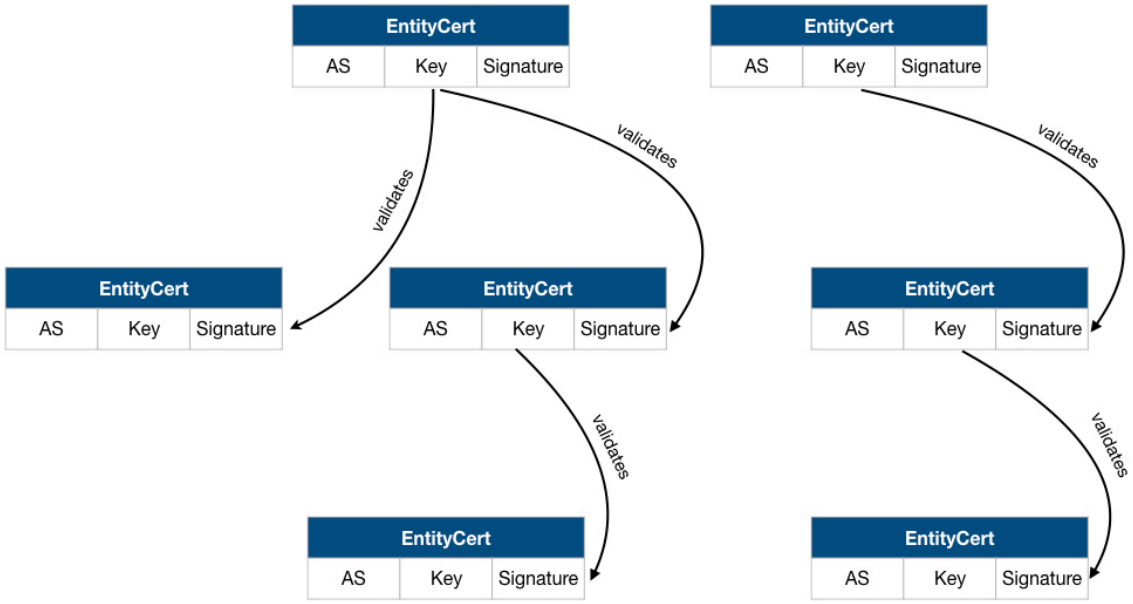

Figure 3.2: Web of trust for entity certificates (based on Figure 1 from [21])

Authorization certificate

Authorization certificate (AuthCert) is used to authorize AS prefixes. Namely, it checks whether an AS is eligible to advertise allocated prefixes. The certificate contains IP prefixes(s), AS number and a public key of certificate owner, AS number and signature of certificate signer.

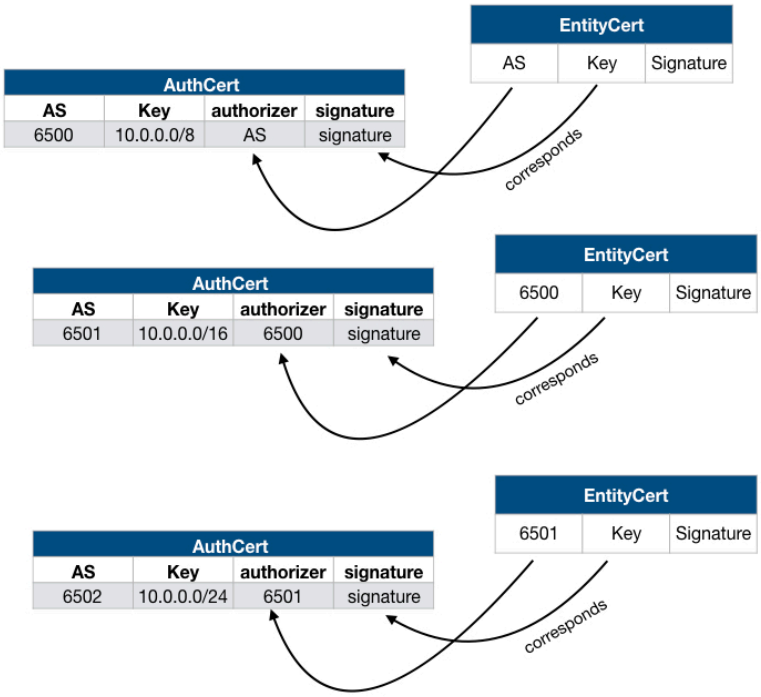

Figure 3.3: Authorization certificate distribution (based on the Figure 2 from [21])

In Figure 3.3 top level AS has authorized AS6500 to use 10.0.0.0/8 prefix. In order to do that, it issued a certificate for AS6500 which is signed by its private key. As a result, AS6500 can assign sub-prefixes to ASes managed by it in a similar manner. For example, AS65001 was authorized to use 10.1.0.0/16 prefix and allocated prefixes with a smaller range to its children. The AuthCert can include multiple prefixes to reduce cryptographic processing requirements and the number of certificates in a network [21].

In order to validate AuthCert, an AS has to look up the public key of certificate signer and verify that the signature of AuthCert is valid. If the signature is correct, then AS is authorized to use prefixes specified in AuthCert [21]. Each soBGP speaking entity has to maintain a local mapping between ASes and prefixes they are authorized to advertise. The table is used to check the receiving UPDATE messages to ensure that AS is authorized to advertise specific prefix.

Prefix Policy Certificate

Prefix Policy certificate (PrefixPolicyCert) wrap AuthCert with network policies applied per prefix. AuthCerts are not advertised independently but within PrefixPolicyCerts [21]. PrefixPolicyCerts are issued by AS which assigns prefixes to another AS and apply rules the receiving AS should follow. ASes can apply any policies to prefixes. For example, the policy can be a list of ASes that must not or must be in the AS Path of routes for certain prefix. However, the problem is to enforce the receiving AS to apply these policies.

AS Policy Certificate

Previous three certificates are responsible for verifying whether an AS is authorized to advertise prefixes, what network policies it should apply and whether the AS number it claims to have actually belongs to it. AS Policy Certificate (ASPolicyCert) is used to verify that the advertised path is valid.

Each AS attached to the network creates an ASPolicyCert which contains a list of peer AS numbers which this AS belongs to. The certificate is signed by AS it is issued by (self-signed). As a result, the topology of the network could be built by verifying the AS Policy certificates.

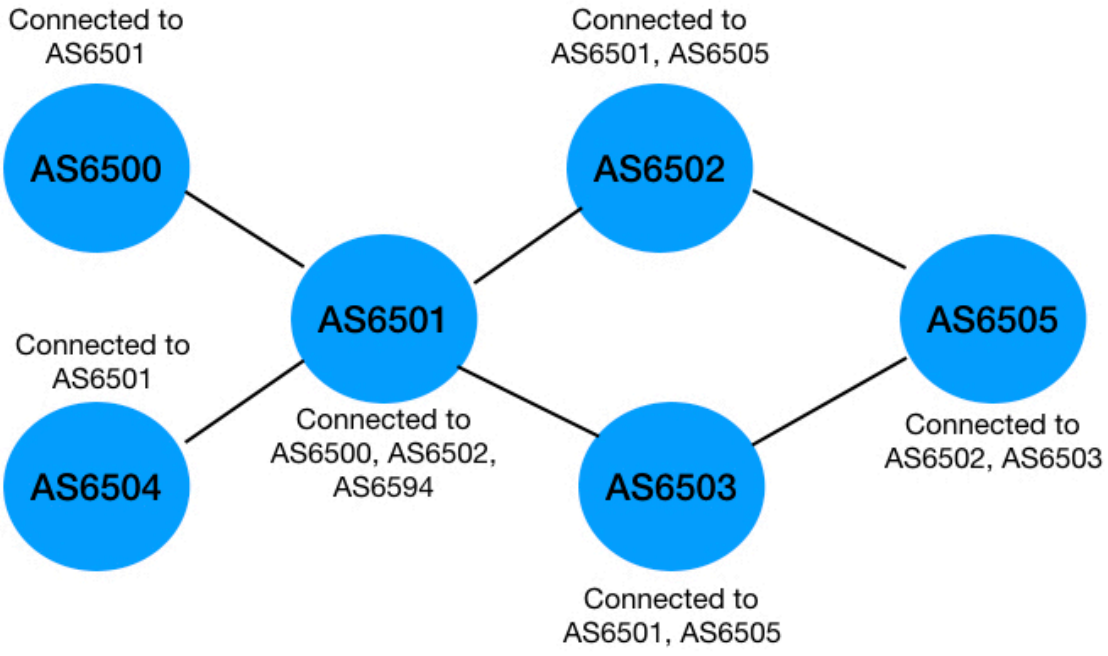

Figure 3.4: Example of AS policy certificate usage (based on the Figure 3 from [21])

In Figure 3.4 AS6505 receives an UPDATE message from AS6502 which claims to have a path to AS6500 through AS6501. AS6505 performs the following checks to verify that the path is valid:

- Check ASPolicyCert of AS6502 to verify that it is connected with AS6501.
- Check ASPolicyCert of AS6501 to verify that it is connected with AS6502.
- Check ASPolicyCert of AS6501 to verify that it is connected with AS6500.
- Check ASPolicyCert of AS6500 to verify that it is connected with AS6501.

In case AS advertises path that is incorrect, the receiving AS can identify the issue by verifying ASPolicyCerts of ASes in a path. In addition, ASPolicyCert can include AS specific statements, etc., a specific peer is not a transit, not advertising certain peers [21].

Certificates management

soBGP has three different certificates (EntityCert, PrefixPolicyCert, ASPolicyCert) which have to be exchanged between peers for path verification. The certificates are exchanged using a new type of BGP messages called SECURITY. The messages can be exchanged in four different ways [21]:

- Certificates are exchanged and validated between peers from different ASes. The exchange is performed by AS border routers which validate received certificates, exchange them with other peers and store in a local database. The database is used to validate incoming UPDATE messages.
- Certificates are exchanged between peers from different ASes, but validation is delegated to other servers within an AS. When routers receive an UPDATE message, they query those servers for the validity of the received PATH.
- Certificates are exchanged over a multihop session. Servers within ASes process the certificates and border routers of different ASes query them to verify the validity of paths in UPDATE messages.

Problems

soBGP tackles not all of the BGP security problems. It has much weaker path authentication compared to S-BGP. The main difference between approaches taken by soBGP and S-BGP is that S-BGP provides dynamic path attestation. It means that S-BGP peers have a real-time view of the topology and path taken by the message. In contrast, soBGP uses databases which provide a static view of the topology and paths in it. The topology

changes can be taken into account by reissuing ASPolicyCert. As a result, when an UPDATE message is received, it could have a path from changed topology which was not yet reflected in the soBGP database (requirements **S-3,4,5,6**).

soBGP requires a database to be deployed on AS side and the certificates distribution could cause some deployability issues (requirement **D-6**). The fact of having an additional message type, SECURITY message, can introduce some difficulties for protocol deployment and backward compatibility (requirement **D-1**) [21].

✗ Security: S-3, S-4, S-5, S-6, S-12

✗ Deployment: D-1, D-6

Summary

soBGP provides more flexible and lightweight solution than S-BGP to tackle BGP security problems. Administrators can configure the protocol in a way that it will trade off security and protocol overhead. However, it does not provide hop integrity, good origin authentication and attacker can still intrude into the path. Also, it has a weak mechanism for protecting path authenticity and PKI keys distribution.

Furthermore, path and policy validation requires additional databases which makes the deployment more complex and it is more difficult to manage distributed trust.

3.3 Interdomain Route Validation

Interdomain Route Validation (IRV) is a receiver-driven protocol proposed by G. Goodell in 2003 [9]. It is the least centralized solution among others for securing BGP [14].

Every AS in a network contains an IRV server (see figure 3.5). Upon reception of an UPDATE message, the BGP speaker will contact IRV server in its AS to verify the correctness of the received message. The IRV server will validate the received request by querying IRV server from AS mentioned in an AS_PATH. In order to validate all ASes in AS_PATH, IRV server will have to query all respective IRV servers [14]. It can be adjusted to query only certain subsets of ASes in the path or cache results for future usage to improve the performance.

Similarly to S-BGP, IRV can use TLS or IPsec for securing TCP/IP communications. As a result, messages exchanged between IRV servers are ensured authenticity, integrity, and confidentiality. IRV servers can tailor responses to queries based on the requesting entities [14]. This makes the IRV to control the exposure of sensitive routing data such as policy and peering relationships.

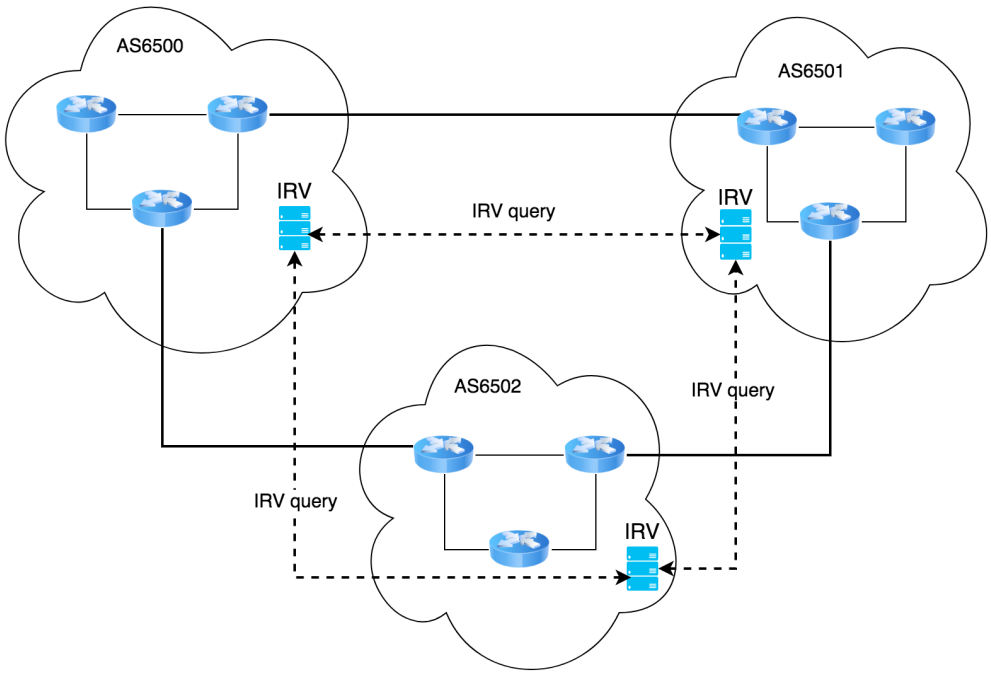

Figure 3.5: IRV servers withing ASes (based on the Figure 6 from [14])

Problems

The main problem of IRV is that it has to have connectivity to IRV servers in other ASes to verify the path [14]. This makes IRV server setup more complicated and maintenance tedious (requirement **D-6**) as it has to keep connectivity with other IRV servers. In addition, deploying IRV server requires separated virtual machine or bare metal server and AS administrators have to make it highly available and resistant to failures (requirement **D-3**).

Moreover, if IRV server will be compromised, it could lead to incorrect behavior of AS and its peers.

✗ Deployment: D-3, D-6

Summary

Unlike S-BGP and soBGP, IRV's operations are independent of the routing protocol [14]. Its security verification is completely detached from BGP protocol which allows more flexibility and makes it backward compatible with BGP. However, there are some issues with deploying and maintaining IRV server which could make this solution not optimal for real-life scenarios.

3.4 BGP MD5 authentication

Message Digest5 (MD5) authentication can be used to secure the connection between BGP peers [1]. In order to do that, peers have to agree on the same password, otherwise, they will not be able to establish a connection. MD5 authentication verifies each segment of TCP connection between peers. Each packet exchanged between peers includes MD5 digest which is calculated by sender [1]. The receiving entity calculates the hash of the packet and verifies that it is the same as included in the packet. As a result, messages exchanged between peers cannot be modified or injected.

- Security: BGP MD5 authentication provides a secure connection between peers, but does not validate UPDATE messages.
- Deployment: BGP MD5 authentication is easy to deploy and to setup. It is a very lightweight solution.

3.5 BGPSec

BGPSec is a standard BGP securing protocol proposed in September 2017 [8]. It is an extension to BGP which makes sure the advertised path between ASes is correct. BGPSec uses RPKI and conceptually is very similar to S-BGP. However, there are some differences compared to S-BGP [8]:

- BGPSec replaces AS_PATH in UPDATE messages with BGPsec_PATH which contains 1 or 2 signature blocks. Usually, one signature block is used. However, for backward compatibility with cryptographic algorithms second block can be used.
- Signatures of ASes in a path contain not only signed AS number but also the AS number of the peer to which the path should be advertised.
- In case peers are not able to communicate in BGPSec, then BGPSec replaces its BGPsec_PATH with AS_PATH of BGP protocol.

Similarly to S-BGP, BGPSec allows malicious peers to remove signatures from the messages and further ASes will not be able to verify them. (requirement **S-12**).

BGPSec eliminates most of the yet known path hijacking attacks on BGP. However, it is still expensive to adopt this protocol and the performance is reduced due to signatures' calculation for each UPDATE message (requirements **D-3,6**).

✗ Security: S-12

✗ Deployment: D-3, D-6

3.6 Secure Blockchain Trust Management system (SBTM)

SBTM can be used to instantiate a blockchain-based PKI for secure BGP modifications [4]. It could be used by ASes to store and share data required by secure routing protocols such as S-BGP, soBGP, BGPsec to validate UPDATE messages. It aims to reduce the operational cost and ease the deployment of secure BGP protocols.

As a result, it can be used in combination with existing BGP secure solutions to make their deployment and management more secure and user-friendly. However, it does not provide any protection against common BGP attacks such as ASN and prefix hijacking.

3.7 Summary

Proposal	Origin authentication				Path authentication			
	Design	Security	Overhead		Design	Security	Overhead	
			Time	Space			Time	Space
S-BGP	Hierarchical PKI local memory	Strong	Low	High	Signatures in messages	Strong	High	High
soBGP	Hierarchical PKI separate database	Strong	Low	Low	Topology map	Low	Low	Low
IRV	Separate IRV servers	Strong	Low	Low	Distributed database	Medium	High	Low

Figure 3.6: BGP secure solutions evaluation

This section reviewed only several secure BGP proposals. However, there were more research performed in secure routing area (e.g. [15]). The reviewed protocols have their own tradeoffs (Figure 3.6) and are not optimal for all the scenarios. For example, S-BGP, BGPsec (has the same results as S-BGP), and IRV have solid security but have some performance and deployment issues. On the other hand, soBGP and MD5 authentication are easier to deploy, but they are lacking some of the security properties or are not backward compatible with BGP. As a result, there is still space to improve and come up with a better, more robust and secure solution.

Chapter 4

Methodology

The smart contracts based design aims to eliminate problems encountered in BGP secure modifications described above. It demonstrates how smart contracts can decouple the information required to validate the UPDATE messages and distribute it among peers in a secure, robust way.

This chapter gives an overview of securing BGP using smart contracts. Firstly, it reviews the main tools used in a described solution such as smart contracts and Quorum blockchain platform. Next, it provides a detailed description of a Registry and AS smart contracts used to check the correctness of UPDATE messages exchanged among BGP peers. Finally, it describes future work that could be done to improve the solution.

4.1 Smart contracts

Smart contracts are programs which execute in a decentralized platform such as Ethereum. They are written in Solidity programming language, which is Turing complete, and are executed inside Ethereum virtual machines (EVM). Before pushing the contract to the P2P network, it has to be compiled and the application binary interface (ABI) has to be extracted. The ABI stores signatures of contract functions and is used to call them and interact with a contract. Before a contract can be used, it has to be mined into a block and synchronized among peers. Each contract change is called a transaction and has to go through the process of validation and adding to block by the blockchain platform. The contracts have a separate storage which is used to store their state. Each change to that state requires to be added to block and synchronized among peers.

When the contract is pushed to the blockchain, it cannot be changed anymore. However, there are mechanisms which allow to upgrade it, e.g. proxy contract. The advantage of using a smart contract is that it cannot be changed by anyone and works exactly the way it was written. They are executed by each peer of the network and their state can only be modified by using functions listed in ABI.

The smart contract has the following features [2]:

- Transparency. All the contracts stored in a blockchain are visible and can be accessed by all the network peers.
- Security. All the transactions are signed by the private key of the issuer. As a result, if a third-party would modify the transaction, then the signature will not be valid anymore. In addition, the blockchain concept allows to store and exchange transaction in a most secure way [6].
- Speed. Blockchain platforms use peer to peer connections to exchange data. As a result, when a new block is added to the blockchain by one of the peers, it gets synchronized among them with no delay.

4.2 Quorum

Quorum [17] is a blockchain platform developed by J.P. Morgan. It is an Ethereum fork which has a different consensus mechanism and contract privacy. The primary extensions of Quorum compared to Ethereum are following:

- Quorum supports transaction and contract privacy. Namely, it is possible to create contracts or transactions which could only be accessed by address owners specified during contract or transaction creation.
- In public blockchains such as Ethereum, anyone can become a peer and synchronize blockchain database. In Quorum it is possible to create a permissioned network. Namely, each node can specify with what peers it wants to synchronize blocks. This way only trusted nodes can become part of the network and communicate.
- Ethereum uses proof of work consensus mechanism. It means that in order to finalize block, mines have to perform heavy computations. Quorum, however, offers Raft-based and Istanbul BFT consensus mechanisms, which are more suitable for permissioned peer to peer network.
- Quorum has significantly better performance compared to the Ethereum.

Permissioned nodes

The Quorum would fit BGP needs much better than Ethereum or any other blockchain platform. It does not require ether (Ethereum cryptocurrency) or mining in order to maintain a consensus of the blockchain. However, each peer in the BGP network has to keep a `JSON` file which maintains a list of nodes in the format:

```
[
    "enode://remotekey1@ip1:port1",
    "enode://remotekey1@ip2:port2",
    "enode://remotekey1@ip3:port3",
]
```

Listing 4.1: Quorum node permissioned nodes list

The list specifies what nodes can connect to a given node and also to which nodes the given node can dial out to. It should be specified when starting the node using `--permissioned` flag. This way blocks will be only synchronized among permissioned nodes. For example, if AS 1 is connected to AS 2 and 3, then it has to specify enodes of those ASes and enode of IANA. From each AS it is connected to, it has to retrieve remote key (generate at node startup), IP address and port (usually all the nodes have the same). When AS connects to the new peer, it can add the enode of this AS dynamically without rebooting the node. In addition, if peer disconnects from current AS, then its enode can be removed from the list.

Another way of modifying the list of permissioned nodes is to use JS console. An admin can issue `raft.removePeer(raftId)` to remove the node from peers, where `raftId` is the number of the node you wish to remove. To add a node to the list of peers, issue `raft.addPeer(enodeId)`, where `enodeId` is in format `remotekey@nodeIP:nodePort`. The drawback is that each AS has to keep a list of nodes it wants to synchronize with. In the future release of Quorum, the list of nodes will be moved to a smart contract and would not require AS to keep this list.

Consensus mechanism

Quorum has two different consensus mechanisms: Raft-based and Istanbul BFT. Both of them have their own advantages and disadvantages, but, in general, they are quite similar. The Raft-based consensus mechanism has got more popularity in Quorum community due to its performance and block addition mechanism which ensures there will not be forks in a blockchain. In Raft-based consensus mechanism, Byzantine fault tolerance is not a requirement due to the permissioned list of peers who maintain the blockchain.

There are several differences of Raft consensus mechanism compared to traditional proof of work used in Ethereum:

- In Raft miners do not submit blocks simultaneously, instead, there is one leader which is elected using Raft protocol. It mines the transactions and submits them into blocks. It is called a leader.
- The leader does not need to perform heavy computations and provide proof of work, instead the transactions are submitted immediately. This allows Raft-based consensus to gain significant performance compared to the Ethereum proof of work.

- When a leader is about to submit a new block, it first shares the block with peers and the block is applied synchronously. As a result, there will be no forks that could occur in proof of work when multiple miners submit blocks.
- Raft-based consensus has a speculative mining mechanism which optimizes synchronization and simultaneous insertion of the new blocks into the chain. It makes the process of transaction propagation across the network faster.
- If the leader is not emitting new blocks or behaves incorrectly, it stops to be a leader and the network elects a new leader using Raft protocol.

4.3 Solution overview

BGP UPDATE messages can be validated using smart contracts running on a blockchain platform. When an UPDATE message is received by AS, it could make a query to a blockchain to validate attributes of the UPDATE message. In order to interact with a blockchain and participate in a peer to peer network, each AS border router has to run a blockchain node.

There are two types of smart contracts: Registry and AS. Each AS after registering with IANA receives a smart contract which is used by itself and other ASes to validate connections between ASes in `AS_PATH` of the UPDATE message. Registry contract is used to keep track of prefix allocations, ASes, and their contract addresses. It can be used to retrieve the contract address of AS or to validate whether prefix belongs to specific AS.

In the following sections, registry and AS smart contracts will be described in more details. In addition, accounts management used to make transactions into the blockchain, will be explained. Furthermore, it will be described how exactly `AS_PATH` validation is performed and how contracts could be extended, improved in the future.

4.4 Account management

Each smart contract state modification is represented as a transaction. The transaction has to be signed by account's private key. As a result, in order to interact with a blockchain, an account has to be generated. Each account is represented as a public, private key pair. The address of an account is a public key which went through multiple hashing algorithms. It can be shared with anyone and is used to distinguish entities in a blockchain. When the contract is created, its address is generated based on the public key of an issuer and additional nonce.

The private key of the contract is only used to sign transactions and should never be shared with anyone. It is usually protected with a passphrase and stored in a most secure, private way. The private key should be unlocked before making any transactions to the

blockchain. A user has to specify passphrase and, optionally, the time period the accounts should be unlocked for.

There were multiple CLI tools developed by Ethereum community to interact with a blockchain such as `web3` and `geth`. These tools can be used to generate accounts. The Appendix A includes Python script which uses `web3` and `geth` to create, list and unlock accounts (see Listing 4.2).

```
1 $ python3 account.py --ipc-path=$IANA_IPC create owner secretPassphrase
2 Account with name owner and address 0
      xa2b03306C6074Be3e834F39b24b8BB6be8752A45 has been successfully
      created.
3 0xa2b03306C6074Be3e834F39b24b8BB6be8752A45
4
5 $ python3 account.py --ipc-path=$IANA_IPC list
6 owner: 0xa2b03306C6074Be3e834F39b24b8BB6be8752A45
7
8 $ python3 account.py --ipc-path=$IANA_IPC unlock owner secretPassphrase
9 Account with name owner has been successfully unlocked.
```

Listing 4.2: Create, list and unlock ethereum account

4.5 Registry contract

A "Registry" is a name for the smart contract which can be used by IANA to keep track of ASes and prefix allocations in a network. In addition, ASes and other network entities could use the Registry contract to validate whether specific prefix was allocated to the AS or to receive the address of an AS contract (explained below). In the following sections, IANA will be referenced as a Registry contract owner. However, the Registry contract could be created by any node in a blockchain network.

In order to deploy a Registry contract, the following steps must be performed:

1. IANA creates an account (keypair) for making transactions in a blockchain.
2. IANA issues a contract deployment transaction which is signed by a private key of the account created in a previous step.
3. Blockchain node adds a transaction to the transactions pool and it gets synchronized among network peers.
4. When transactions will be mined (added to the block and included to the blockchain) by one of the peers, the new block will be synchronized among peers.

IANA has to create an owner account before deploying the contract (see 4.4). Only Registry contract owner can modify the Registry contract state. Note, that the private key used

to sign the Registry contract deployment transaction should be stored in the most secure way and only be used when creating other transactions for the Registry contract. If the private key is compromised, the deployed Registry contract cannot be trusted anymore. The account address (public key) is saved during contract initialization:

```
address public owner;

constructor() public {
    owner = msg.sender;
}

modifier onlyOwner() {
    require(msg.sender == owner);
    _;
}
```

Listing 4.3: Registry contract owner initialization

In Listing 4.3 the modifier is created for registry owner. This modifier could be used in other functions to allow their executions only for registry owner. For example, function with definition `function assignPrefix() external onlyOwner {}` will only be executed in case transaction was signed by private key which corresponds to a public key (address) stored in `owner` variable.

When the Registry contract is created, it will be assigned a unique address. The address is a unique identifier for smart contracts running in a blockchain. The contract address is computed from the public key of the contract owner and the number of his transactions in a blockchain. The address should be stored somewhere and used to identify the registry. If the address of a registry is lost, it could be recovered by checking transactions in a blockchain executed by the owner. Each blockchain peer is able to access a registry from its local copy of a blockchain.

AS registration

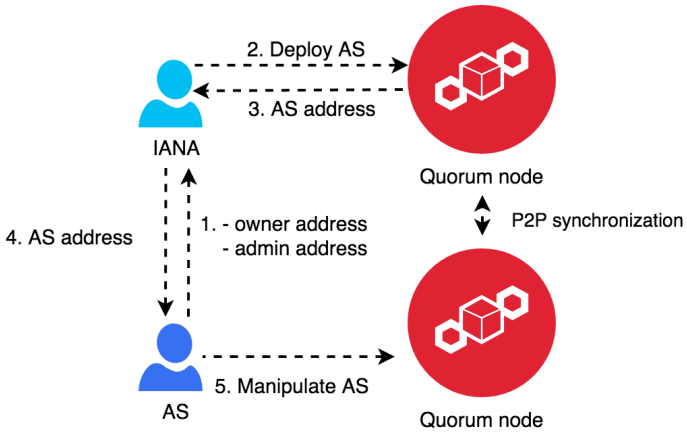

Figure 4.1: AS creation by registry owner.

When the Registry contract is deployed, IANA can register ASes (see Figure 4.1). In order to do that, each AS has to generate an owner account (see 4.4) which they could use to interact with their AS contract in future. When AS generates this account, it shares an account address (public key) with the IANA. After receiving an owner address, IANA generates a unique AS number and creates a transaction to the Registry contract by calling `createAS` contract function:

```
1  mapping(uint32 => address) private ASes;
2
3  function createAS(uint32 number, address ASowner) external onlyOwner {
4      require(ASes[number] == address(0));
5      ASes[number] = new AS(ASowner);
6  }
7
8  function getAS(uint32 number) public view returns (address) {
9      return ASes[number];
10 }
```

Listing 4.4: AS contract creation

The Registry contract stores a mapping between an AS number and its contract address. Therefore, any blockchain network participant can retrieve an AS contract address by using `getAS` function and specifying its number. The AS contract is used by AS to store the information that it wants to share with other ASes. The AS contract will be described in more details in following sections.

Prefix allocation

IANA can use a Registry smart contract to assign and remove prefixes.

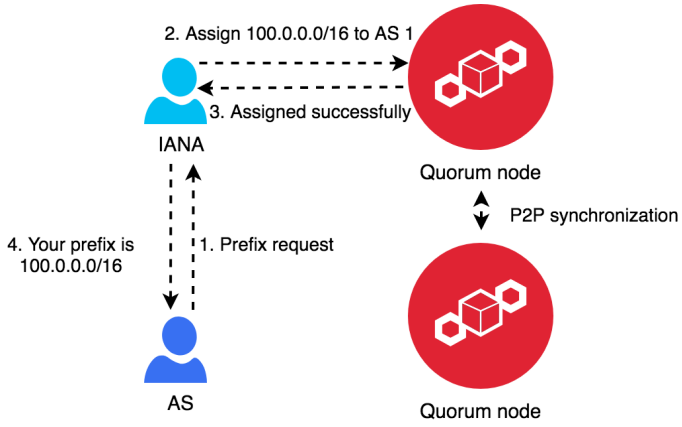

Figure 4.2: Prefix allocation to AS by registry owner

After processing a request from AS to allocate prefix to it, IANA makes a request to registry smart contract by calling `assignPrefix` function and specifying AS number and prefix (see Figure 4.2). The prefix should be available, otherwise, the transaction would be reverted. When a prefix is allocated to AS, it is saved in a mapping of the Registry contract (see Listing 4.5) and synchronized across all the peers. Other ASes can check whether prefix really belongs to a certain AS by calling `hasPrefix` function and specifying AS number and prefix they want to check a connection between. When IANA would like to remove the prefix from an AS, it will issue `removePrefix` contract function and the connection between AS and prefix will be removed (see Listing 4.5).

```
1  mapping(bytes32 => uint32) private prefixes;
2
3  function assignPrefix(bytes32 prefix, uint32 asNumber) external
       onlyOwner {
4      require(ASes[asNumber] != address(0) && prefixes[prefix] == 0);
5      prefixes[prefix] = asNumber;
6  }
7
8  function removePrefix(bytes32 prefix) external onlyOwner {
9      delete prefixes[prefix];
10 }
11
12 function hasPrefix(bytes32 prefix, uint32 asNumber) external view
       returns (bool) {
13     return prefixes[prefix] == asNumber;
14 }
```

Listing 4.5: Prefix allocation to AS

4.6 AS contract

An AS contract represents each autonomous system in a BGP network. The contract must be created by IANA which registries new AS in a network (see 4.5). As a result, each AS in a BGP network has its own contract in a blockchain.

```
1  contract AS {
2      address private owner;
3      address private admin;
4      address public registry;
5
6      bool private disabled;
7      mapping(uint32 => bool) private peers;
8
9      constructor(address _owner) public {
10         owner = _owner;
11         admin = _owner;
12         registry = msg.sender;
13         disabled = false;
14     }
15 }
```

Listing 4.6: AS contract constructor

When the AS contract is created by IANA, its `constructor` (see Listing 4.6) is called with the `owner` address which was received by IANA from the AS. The constructor sets permanently an owner address and initializes other variables which are described in following sections. The IANA address is also stored in a contract in `registry` variable. It is a public variable and can be used by AS to retrieve information about prefixes and other AS addresses from the IANA registry.

An AS has to store the address of the created contract in order to interact with it. If AS has lost its contract address, it can retrieve it by calling `getAS` of the registry contract and specifying its AS number.

Permissions

Each AS contract stores information about two types of users: `owner` and `admin`. The owner is initialized during contract creation and is usually passed by IANA. This user has the highest permission rank in the contract. It can execute all the contract functions:

- Change `admin` account address.
- Disable and enable AS contract.
- All the action that could be performed by `admin` account.

The owner contract should be stored in a most private way and used only for extreme actions such as first two listed above. For all the other actions `admin` account should be used. The contract functions allowed only to `owner` include following modifier:

```
1  modifier onlyOwner() {
2      require(msg.sender == owner);
3      _;
4  }
```

Listing 4.7: AS owner modifier

The `admin` account should be used on a daily basis to add and remove peers to/from the contract (described in details below). All the functions which can be executed by admin are protected with following modifier:

```
1  modifier onlyAdminOrOwner() {
2      require(msg.sender == admin || msg.sender == owner);
3      _;
4  }
```

Listing 4.8: AS admin modifier

From the modifier in Listing 4.8 it could be seen that all the functions which could be executed by admin can also be executed by the owner. Note, that `owner` and `admin` accounts should be stored on separate machines for security reasons.

There is also a number of read-only functions described below which can be accessed by any peer in a network. They do not have any modifier applied to them.

Peers

Each AS in the BGP network has peers directly connected to it and exchanging UPDATE messages. The routing between AS and its peers is set up manually by AS administrators. In order to make other ASes aware of connections between peers, each AS keeps track of AS numbers it is connected to its smart contract. As a result, peers can query the AS contract to verify the path in an UPDATE message.

Peers can only be modified by `admin` or `owner` account. Other accounts can only verify whether specific AS number is in a list of AS's peers in its contract.

```
1  mapping(uint32 => bool) private peers;
2
3  function addPeer(uint32 number) external onlyAdminOrOwner {
4      peers[number] = true;
5  }
6
7  function removePeer(uint32 number) external onlyAdminOrOwner {
8      delete peers[number];
9  }
10
```

```
11 function hasPeer(uint32 number) external view returns (bool) {
12     return peers[number];
13 }
```

Listing 4.9: Adding and removing AS peers

In order to add a peer to a list of peers in AS contract, AS admin has to execute the `addPeer` function. After that, the AS number will be stored in a mapping `peers`. All the other peers could verify whether the AS claims to be connected to another peer in a network, by issuing `hasPeer` function. The connections between two peers could be considered valid if both of them claim to be connected in their smart contracts.

If the connection between peers is not valid anymore or there are connectivity issues, the AS could issue `removerPeer` function to remove peer from peers list. The mapping will be updated respectively and all peers will be aware of that.

Enabling and disabling AS contract

An AS owner can disable or enable contract when needed. There can be several reasons to do that. For example, an AS could be disabled in case it was compromised (see Figure 4.3). In order to prevent a malicious entity from adding invalid AS numbers to peers mapping in contract or listening to traffic, an AS owner can disable AS contract. As a result, all its peers could see that change from the blockchain and redirect traffic from it to different paths. When AS would get into normal state, an AS owner can make a query to change the contract admin address, enable AS and the traffic can be traversed as before.

In addition, AS can be disabled for maintenance. When AS border router is not available, AS can notify other peers to traverse the traffic another way by disabling the contract.

Lastly, when AS wants to not participate in blockchain anymore, it can disable AS permanently, so the peers will not rely on the data in the contract.

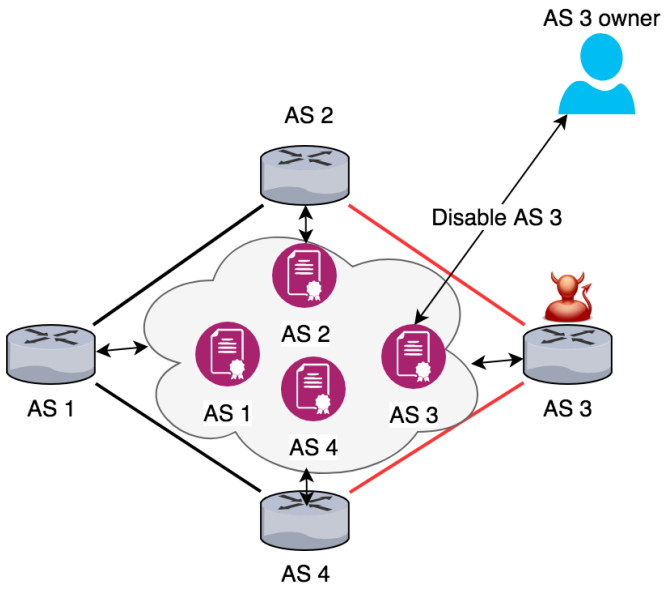

Figure 4.3: AS owner disables AS contract while it is compromised.

An AS contract is enabled by default. In order to disable it, the owner has to call `disable` function (see Listing 4.10). When transaction will be synchronized by peers, they could see that the AS is disabled by calling `isDisabled` function. When AS owner wants to enable AS contract back, the `enable` function should be called.

```
1  bool private disabled;
2
3  function disable() external onlyOwner {
4      disabled = true;
5  }
6
7  function enable() external onlyOwner {
8      disabled = false;
9  }
10
11 function isDisabled() external view returns (bool) {
12     return disabled;
13 }
```

Listing 4.10: Disabling and enabling AS contract

4.7 Path validation

BGP peers use UPDATE messages to inform network entities about prefix reachability. Each UPDATE message contains `AS_PATH` which consists of AS numbers and net-

work layer reachability information which contains the prefix UPDATE message is advertising. For example, AS 4 receives UPDATE message with `AS_PATH=1, 3` and prefix `100.0.0.0/16`. It means that it could send all the packets with IP from prefix `100.0.0.0/16` to AS 3, which in turn will redirect packets to AS 1, the owner of the prefix.

However, the malicious entity could hijack the path or prefix and make AS to apply an invalid route. As a result, AS should be able to validate the received path and prefix in an UPDATE message. It could be done using Registry and AS contracts.

Prefix validation

The first step to validate path advertisement received by AS from its peers is to check whether prefix belongs to first AS in the path. In order to do that, it has to perform the following steps:

1. Get registry address from owned AS contract. An AS has to get a value of `registry` variable from the contract.

2. Use the registry address to locate Registry contract in a blockchain.

3. Call `hasPrefix` function in Registry contract with specifying advertised IP prefix and first AS number in a received path. For example, to check whether AS 3 owns `100.0.0.0/16` prefix `hasPrefix("100.0.0.0/16", 3)` should be called.

4. Based on the output decide whether to reject the `AS_PATH` or continue validation.

Due to the fact that prefixes mapping is stored in registry contract and controlled only by IANA, there is no way any AS can modify it. In addition, peer to peer synchronization allows peers to see the changes in a contract as soon as they will be added to block and synchronized among nodes (see performance evaluation below).

Path validation

After first AS in the `AS_PATH` proved to be the owner of the prefix, there should be a mechanism to validate whether ASes in a path are connected with each other. When AS receives UPDATE messages from its peer, it can validate connections between ASes in a path using the following algorithm:

1. Get registry address from owned AS contract. An AS has to get a value of `registry` variable from the contract.

2. Use the registry address to locate Registry contract in a blockchain.

3. Call the `getAS` function of the registry contract to get the contract address of AS under the review. The AS number has to be specified as an argument. For example, to get the contract address of AS 3, `getAS(3)` should be called. If the contract address was not returned, it means that AS was not registered and path validation has failed.

4. Use the contract address received from the previous step to locate the AS contract in a blockchain.

5. Check whether AS contract from the previous step is not disabled by calling the `isDisabled` function. If the output will be `true`, then validation has failed.

6. Check whether AS numbers in `AS_PATH` which are before and after the AS number under the review, are its neighbors. In order to do that, `hasPeer` function has to be called. For example, if `AS_PATH=1,3,4` and AS 3 is under the review, then both `hasPeer(1)` and `hasPeer(4)` has to return `true`.

7. If the previous step was successful, then continue with next AS number in `AS_PATH` from step 3. Otherwise, the validation has failed.

The described algorithm ensures that both ASes in a path that goes one after another, claim to be connected using smart contracts. For example, path `1, 3, 4` is valid only in a case following checks will succeed:

1. AS 1 contract: `isDisabled()` -> `False` and `hasPeer(3)` -> `True`

2. AS 3 contract: `isDisabled()` -> `False` and `hasPeer(1)` -> `True` and `hasPeer(4)` -> `True`

3. AS 4 contract: `isDisabled()` -> `False` and `hasPeer(3)` -> `True`

This way it can be validated whether both ASes claim to be connected and prevent the case when one of the ASes was compromised and sends malicious UPDATE messages, where it claims to be connected with some other ASes. An AS could also listen to emitted events when new blocks arrive and make relative changes. For example, there could be an event emitted when one of the ASes has removed its peer. Other ASes could check whether this change affects their routing information and apply changes if so.

4.8 Deployment

Quorum nodes can be deployed on the AS border router's side. Specifically, each border router will have its own Quorum node running and will perform paths validations against it. Due to the fact that the node is running on the same machine, path validations will

be fast. In addition, it will increase blockchain security as there will be more peers in a network and harder to perform attacks on a network. However, there are several major drawbacks of such approach:

1. It will increase memory and space usage for border routers. As a result, not all the routers will be capable of running and synchronizing distributed ledger. Furthermore, it would be possible to perform DoS attacks on such border routers.

2. Block synchronization speed will decrease. Namely, each block mined by the leader would reach peers slower with a higher number of peers in a network.

3. It will dramatically decrease maintainability of each AS as each router would have to set up and maintain Quorum node.

Another way of deploying Quorum nodes in ASes is similar to how it is done in IRV 3.3. Namely, each AS could have one or several Quorum nodes running on dedicated servers which will be responsible for syncing the ledger and validating UPDATE messages (Figure 4.4).

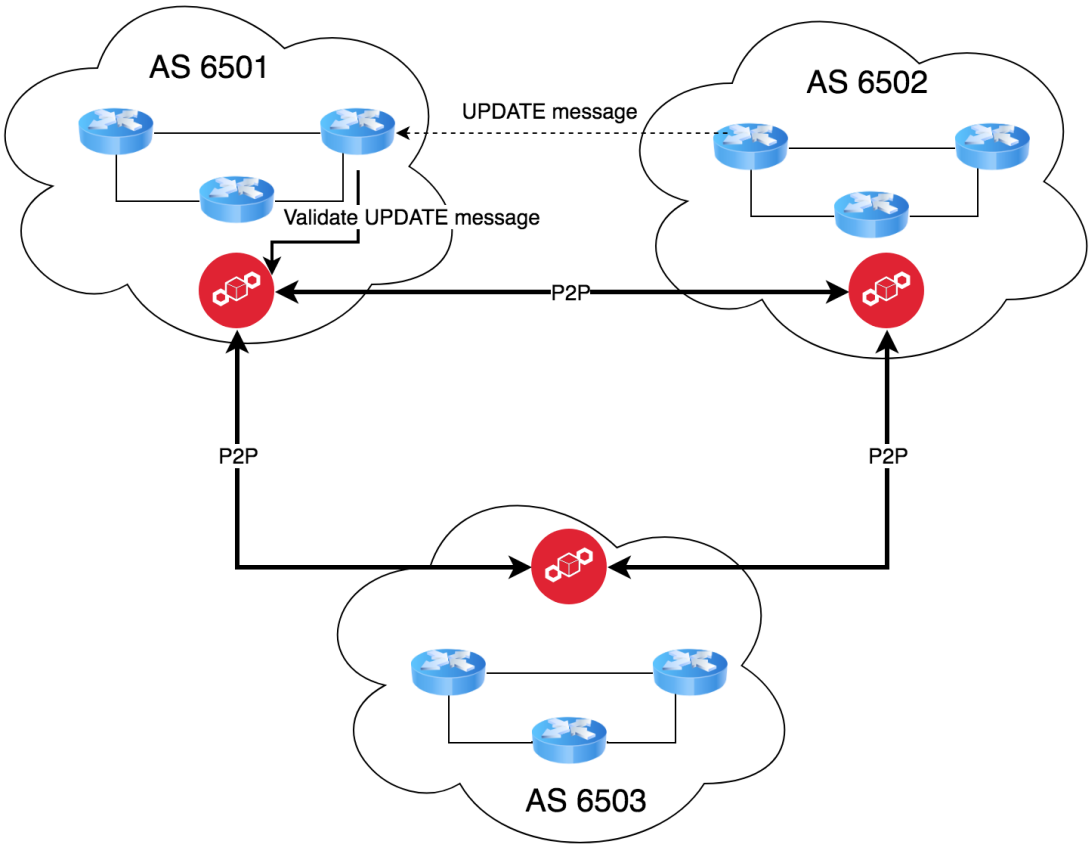

Figure 4.4: Quorum deployment in ASes.

The connection between border routers and Quorum nodes could be secured using existing tools such as IPsec or TLS. In addition, it is possible to deploy multiple Quorum nodes to make them highly available and load balance the traffic. As soon as the Quorum leader will mine the block, it will be synchronized among peers (ASes) and applied to the local copy of the ledger simultaneously. As a result, there is no possibility of having a fork inside the blockchain as well as a race condition.

The procedure for the border router to validate the new UPDATE message will be the following:

1. Border router receives an UPDATE message from the peering AS.
2. Border router establishes the secure connection with the Quorum node.
3. Border router makes a request to Quorum node to validate the UPDATE message.

4. Quorum node performs UPDATE message validation as described in 4.7.

5. Quorum node returns the response to the border router.

6. Border router decides whether to update routing information or not based on the response.

Lastly, it is possible to configure Quorum nodes in a way that they will track changes in a ledger and push updates to the border routers that would be affected by such changes. However, such an approach is quite complicated and requires further investigation.

4.9 Future work

Currently, smart contracts include only the bare minimum required for validating `AS_PATH`. It is possible to extend them with more functionality and, as a result, provide a more secure and flexible solution. For example, Registry and AS contracts does not support upgrading. Namely, it is impossible to fix bugs in them or extend with functionality dynamically. The proxy contract can be used which will track the latest contracts and redirect requests to them.

In addition, all the transactions that change the state, such as adding a prefix to AS or AS is removing its neighbor from the contract, could emit events. Those events will appear on each blockchain node as soon as they will synchronize the block. As a result, they could monitor those events and make certain routing changes if any of the events are related to them.

Furthermore, it is a good practice to document all the lines in a contract. It would provide users a better understanding of how they work.

Registry contract

The current implementation supports only top-level registries such as IANA. However, ASes could also deploy their own Registry contracts and allocate prefixes to their customers. It will reduce load from the IANA registry and ASes could allocate smaller prefixes from prefix allocated to them by IANA to their own customers. One of the possible solutions for that could be the following:

- Each registry stores list of siblings. Those are other registry addresses which should be passed to each AS constructor. For example, if AS 1 has its own registry, its sibling will be IANA registry address. As a result, when AS contract will be created both IANA and AS 1 registry addresses will be passed to its constructor.

- AS constructor should take a list of registries as an input and save them in a `registries` variable.
- When AS receives a new UPDATE message, it will try to find the address of the AS contract from all the registries specified in `registries` variables and fail only if none of them has registered AS in a path.

In addition, Registry contract does not check whether prefixes it wants to allocate are not part of another prefix. For example, currently, prefix `100.0.0.0/16` and `100.0.0.0/24` are considered by IANA registry different. However, prefix `100.0.0.0/24` is part of `100.0.0.0/16` prefix and should not be assigned to any AS.

Furthermore, registries could have the same owner, admin permission model as ASes. It will make them more resistant to the compromisations.

Lastly, Registry contract could distinguish between ASes which have set up their infrastructure to run a node and maintain their smart contract and the ones which have not. As a result, if the AS has not set up the Quorum node yet, its validation will be skipped by other ASes.

AS contract

Currently, AS only include information about their neighboring ASes in order to validate `AS_PATH`. However, they could specify additional information which could be used by other ASes. For example, they could specify network policies for the prefixes they are advertising or default gateways for different prefixes.

Chapter 5

Analysis

This chapter evaluates the security, deployment and performance properties of a solution described in a previous chapter.

5.1 Security

The security of blockchain based BGP is validated against twelve requirements specified in Section 2.2.
S-1: The BGP announcement receiver must be able to determine that the first AS in the received path was authorized to announce the prefix. Namely, it should be able to check that there is a connection between the first AS number and IP prefix in a path.
Solution: In order to verify whether the first AS in a path is the owner of the prefix, the UPDATE message receiver will make a query to a Registry contract (see Section 4.7) and check whether the prefix belongs to the AS number of the first AS in the received path.

S-2: Replay of BGP UPDATE messages should not be possible. However, there could be some mechanism for setting up a window during which the message is still valid.
Solution: When a malformed AS is performing a replay attack on an AS border router by suppressing the valid UPDATE messages and resending the ones which were valid some time ago, it will not affect the victim AS. Due to the fact, that the victim AS will validate the path specified in an UPDATE message by querying the AS contracts in a ledger (see Section 4.7) and verifying that received the UPDATE message is not valid anymore.

S-3: The secure version of BGP UPDATE messages should provide up to date information about paths between ASes.
Solution: All the latest information about AS connections are accessible through the ledger. When any AS has modified its peers in a contract, the transaction is immediately synchronized through the peer to peer network and when the leader has announced a new

block, it will be applied by all the peers in a networks simultaneously. Namely, the change in a topology (AS peers modification) will be known by all the ASes at the same time.

S-4: The secure version of BGP UPDATE messages should provide up to date information about prefix allocations.
Solution: Similarly to S-3, when IANA assigns or removes prefix to/from AS in Registry contract, this information will be immediately synchronized through the ledger and known by all the network participants.

S-5: The secure version of BGP should be able to verify whether an applied path is still valid.
Solution: Each AS stores a blockchain copy, which can be queried to verify whether the path is still valid. Namely, AS can periodically check paths in a routing table and remove the ones which are not valid anymore.

S-6: The secure version of BGP should be able to verify whether the applied path prefix still belongs to the same AS.
Solution: Similarly to S-6, each AS stores a blockchain copy, which can be queried to verify whether the prefix is still valid. Namely, AS can periodically check advertised prefixes in a routing table and remove the ones which are not valid anymore.

S-7: The secure version of BGP should provide link layer integrity between peers. Namely, packets injection, deletion, modification or replay should be prevented.
Solution: This solution does not modify the BGP protocol, but works as an addition to it for verifying paths correctness. As a result, the connections between AS border routers should be secure using existing solutions. For example, BGPsec or MD5 authentication could be used to improve link layer security of peers. The integrity between Quorum nodes is achieved by the fact that blocks are synchronized from different peers and each block correctness is verified upon receiving it. Namely, cryptographical algorithms are used to verify that the received block was not modified.

S-8: The secure version of BGP should reveal information to peers only required to ascertain the correctness of messages. As a result, it should reveal peering, customer/provider relationships as less as possible.
Solution: The information of the peers' connections and prefix allocations is shared among all the Quorum network participants. As a result, the information between peers connections and customer/provider relationships are revealed. However, Quorum project improves the privacy of transactions shared between peers by introducing private transactions which can be read only by peers specified in the `to` field of the transaction. There have to be some improvements done before this feature would be production ready.

S-9: The secure version of BGP should signal or emit logs about security exceptions which are important for network operators.
Solution: Smart contracts allow to emit events when some action was performed on a blockchain. These events will appear on each node as soon as the block will be synchronized. As a result, the Quorum node can trigger different actions based on the events emitted. For example, if peering AS has changed its neighboring peers, the paths which include this AS could be removed or validated again.

S-10: The storage of routing information database should be secured by authentication and imply periodic reauthentication.
Solution: Blockchain is used as a database to store the information about prefixes and peers connections. In order to make changes to this databases, the private and public key pair must be used. The private key is protected with a passphrase and can be unblocked only for a certain period of time. After the unlocking time has ended, the user has to resubmit the passphrase.

S-11: The secure version of BGP should use secure cryptographic algorithms.
Solution: Quorum is Ethereum fork and uses Elliptic Curve Cryptography (ECC) for signing transactions. The cryptography algorithms are considered to be secure for now and are widely used in TLS and blockchain platforms. However, quantum computing is advancing and these algorithms could have to be replaced with quantum resistant algorithms.

S-12: The secure version of BGP should be resistant to downgrade attacks.
Solution: This solution does not modify the BGP protocol and is used as an additional tool for verifying BGP paths. As a result, it is not possible to downgrade this solution to an insecure version.

5.2 Deployment

There are six deployment requirements defined in Section 2.2 which will be used to evaluate the solution.
D-1 The secure version of BGP must be backward compatible with BGP and other versions of secure BGP so that it could be deployed incrementally.
Solution: The blockchain based solution can be deployed incrementally as it does not modify the BGP. However, ASes which use the solution have to distinguish between ASes which have the contract and are maintaining it and ASes which does not have it yet. The Registry contract could store such information and, as a result, other ASes would know whether to validate the AS contract or not.

D-2 The secure version of BGP must be backward compatible with the way messages are formatted, processed and transmitted, so they could be parsed in environments with mixed BGP versions.
Solution: The proposed solution will work with any modification of BGP, e.g., BGPsec, S-BGP and soBGP.

D-3 The secure version of BGP should not possess large memory, storage and CPU overhead on routers.
Solution: If Quorum node was deployed on the same router, it will increase the usage of storage, memory, and CPU. However, if the recommended deployment way is used, then it will not make any impact on the router's resources. However, the processing time of the messages could increase and, as a result, will cause more usage of a router's memory to store the received UPDATE messages which have to be validated.

D-4 The secure version of BGP should allow peers to configure the use of the security mechanism on a per-peer basis.
Solution: Each peer can deploy the Quorum node in any way suitable for its needs. In addition, each peer can regulate what secure information to expose to other peers through its AS smart contract.

D-5 The secure version of BGP should allow peers to apply their custom routing policies on received paths to determine the preferred one.
Solution: When AS has sent path validation of the UPDATE message to the Quorum node and the specified path is considered to be correct, it can apply any routing policies on it locally. Furthermore, AS contract could be extended in a way that each AS would be able to specify what network policies should be applied to its prefixes by other ASes.

D-6 The secure version of BGP should be easy to deploy, maintain and remove.
Solution: In order to run the Quorum node, an AS administrator has to set up a dedicated server, download Quorum package, configure and run it. The setup does not require a lot of time and does not differ a lot from setting up any other software package. The node has to have enough memory, storage, and CPU allocated to make it able to store, process and send the blockchain packets. Also, the administrator has to keep the AS smart contract up to date and make changes anytime there has been some change made in its topology, e.g., it has added/removed a peer. In order to remove the Quorum node, the AS owner has to disable the AS contract and stop running the node.

5.3 Performance

BGP traffic

In order to verify whether the solution will be capable of handling BGP traffic and will not affect network performance, the traffic of BGP speaker has to be analyzed to extract the amount and properties of data being transferred between peers.

The measurements were conducted on AS131072 by Geoff Huston [13]. Its traffic was collected and analyzed since the year 2007. The BGP speaking router was logging all the received BGP UPDATE messages and was not emitting any new ones. It was connected to the default-free eBGP feed from AS4608 in Australia and AS4777 in Japan for both IPv4 and IPv6 routes [13].

As a result, the AS was working as a stub at the edge of the Internet. Its purpose was just to collect and analyze packets from BGP speaking routers.

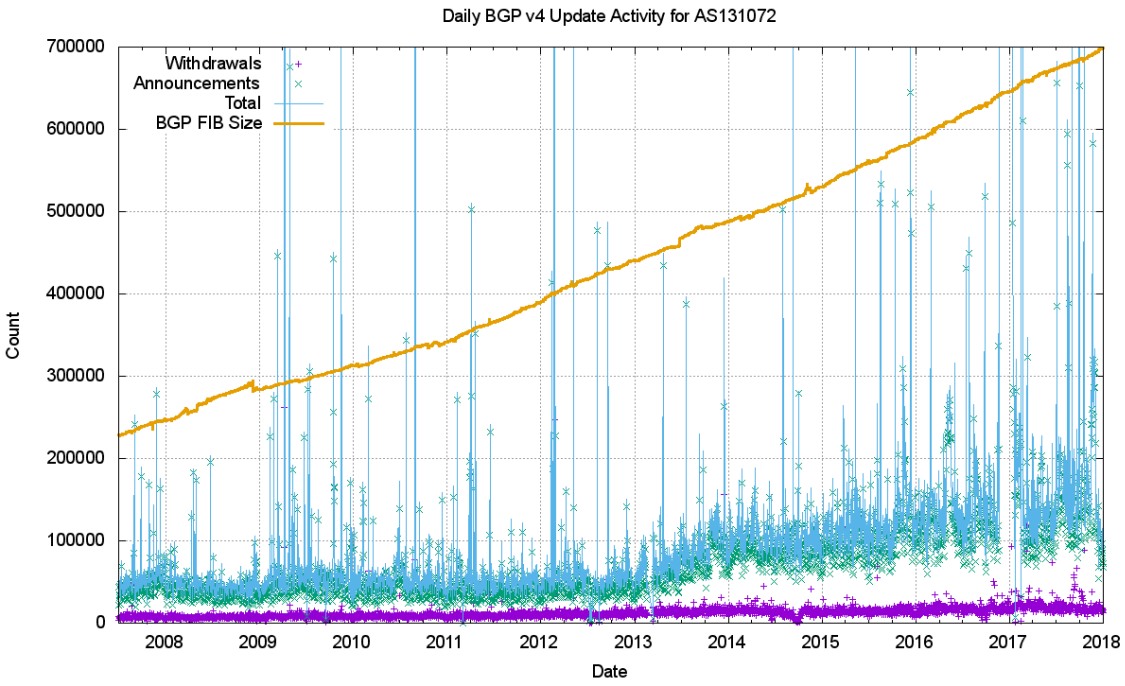

Figure 5.1: Daily BGP v4 Update activity for AS131072 [13].

Figure 5.1 shows the daily BGP v4 activity at AS131072 from June 2007 to January 2018. Following facts could be extracted from it:

1. The number of advertised prefixes has risen from 230 000 to 700 000 by the end of 2017.

2. The number of withdrawals has stayed relatively constant (15 000 – 20 000 withdrawals per day).

3. The number of announcements has reached around 120 000 a day by the end of 2017.

4. The total number of UPDATE messages was steady at 50 000 per day from 2007 to 2013. From 2013 to 2015 the number of UPDATE messages per day has increased to about 100 000. In a period of 2016 to 2017, the number of updates per day has increased again and reached 170 000 UPDATE messages per day by the end of 2017.

Another important fact to take into account is the length of AS Path in UPDATE messages. From Figure 5.2 it can be seen that the average AS Path has remained relatively constant at 5.7. As a result, path validation requires to do queries to blockchain for around 5.7 ASes.

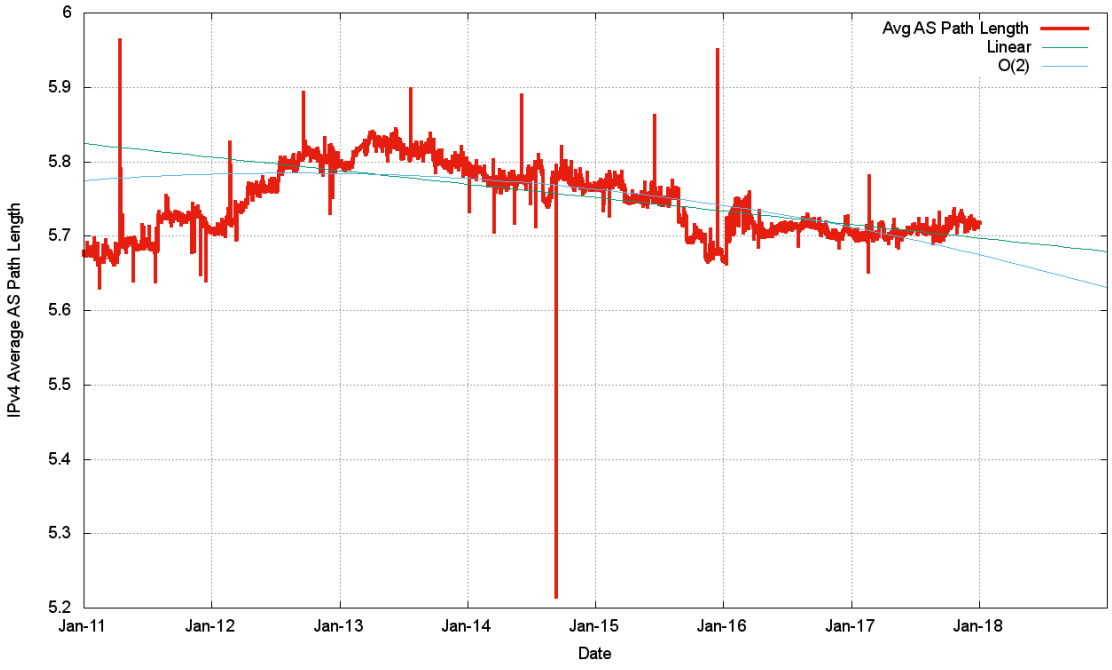

Figure 5.2: IPv4 average AS Path length [13].

Furthermore, the number of ASes in BGP network is constantly increasing (see Figure 5.3). By the beginning of 2018, the number has reached 60 000. However, the announcements are mostly generated by 20 000 - 40 000 ASes [13]. Other ASes are not upstream to any third party. Namely, they do not pass UPDATE messages to other ASes and do not have a large set of BGP peers [13].

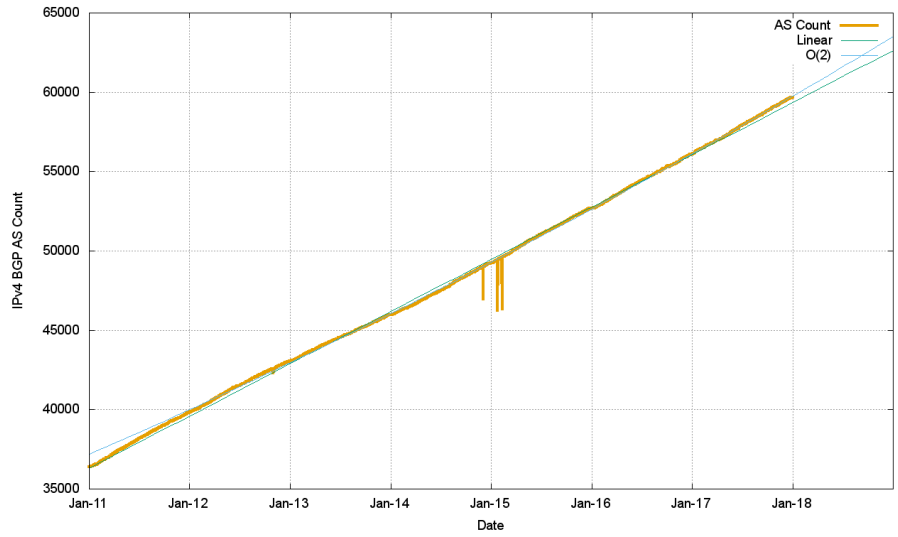

Figure 5.3: The number of ASes in BGP network [13].

As a result, by the January 2018, there were 20 000 - 40 000 ASes which altogether generated around 170 000 UPDATE messages a day. In case of announcements, each UPDATE message had around 5.7 ASes in AS path.

Quorum

After analyzing information about BGP traffic load, it is time to measure whether Quorum will be capable of dealing with such amount of traffic.

Read operations

The actions like UPDATE message validation are read operations and do not perform any changes on a blockchain. In average, each UPDATE message validation should perform around 6 queries to the blockchain, because each AS Path contains 6 ASes in average which should be validated separately. Such queries do not perform any network requests and are executed locally by parsing the blockchain. As a result, they are relatively fast and any modest Desktop PC will be capable of running such node. If Quorum node is deployed on a dedicated server, its performance could be adapted by scaling vertically.

For example, adding more cores, RAM to the server or having SSD storage could make queries execute faster. The amount of resources required depends more on the number of requests AS gets from the border routers.

Write operations

Based on a BGP traffic investigated above, there were around 170 000 UPDATE messages generated a day by the beginning of 2018. Namely, the Quorum network should be capable of dealing with around 2 transactions a second. There should be also some additional margin in case the number of UPDATE messages will grow in the future. It should also be capable of dealing with traffic spikes which could increase up to 4 transactions a second.

In order to apply a transaction to the blockchain, the following steps have to be executed:

1. The transaction has to be propagated to the leader. The leader is a Quorum node which is chosen by peers using a voting system (see section 4.2).
2. The leader has to mint the block. This is performed each 50 ms by default. However, this parameter is configurable and could be adapted to the network needs [18]. This rate is used to achieve a balance between transaction throughput and block propagation time.
3. The leader has to publish the block to the blockchain network through the Raft protocol.
4. The published block has to propagate to all the nodes in a network.
5. The block has to be appended to the existing ledger by all the nodes in a network.

The actions performed in steps 2, 3 and 5 are very fast and could be omitted in calculations. The most time-consuming operations from above are in steps 1 and 4.

The transactions generated by AS and Registry contracts take around 146 000 gas. The gas is an execution fee that transaction senders have to pay for making it executed on a blockchain. However, it is not used in Quorum blockchain and can only be used to calculate the time required to execute the transaction. Currently, Ethereum main chain has 8 000 000 gas limit (regulates the number of transactions per block), 20 000 nodes and 14.6 seconds block propagation time [3]. The connectivity between nodes in a network is varying and machines could be laptops, PCs or dedicated servers. In spite that fact, such a network would be capable of dealing with 54.8 Registry and AS transactions in a second which is 27.4 times more than the BGP load recorded at the beginning of 2018. ASes are even better interconnected than peer to peer Ethereum network and will have even smaller block propagation time.

Quorum is an Ethereum fork. As a result, it would be capable of dealing with the BGP UPDATE messages if there would be 20 000 Quorum nodes in BGP network. If the number of Quorum nodes will increase, it could take more time to propagate the messages to all the nodes and, as a result, decrease the block time.

According to the research performed by Decker and Wattenhofer in Zurich [5], the two main factors which influence the block propagation time are block size and network diameter.

The network diameter depends on the way nodes are connected. The blocks are propagated faster in a dense network. The nodes which have long shortest path will receive the block the last. Due to the fact that RIRs have run out of IPv4 address blocks [16], the BGP network is becoming denser and is not growing anymore in terms of average AS Path [13]. As a result, the BGP peers interconnectivity will increase and the block propagation will not slow down with time.

The block size depends on the total amount of the gas required to execute all of the transactions in a block. With the gas limit of 8 000 000, each block is 22.3 KB in size. Currently, transactions in Quorum are minted every 50 ms and there are 2 transactions per second on average. It could be assumed that each block contains one transaction on average. As transactions generated by AS and Registry contracts take in average 146 000 gas, each block will be around 400 bytes. As a result, the block should be propagated much faster than 14.6 seconds as the block size is 55 times smaller and the proposed solution has a potential of dealing with the current amount of BGP traffic with a large margin.

Chapter 6

Conclusion

In this thesis, it was analyzed how different BGP securing solutions tackle security and deployment issues. It was proven that none of them can address all of the issues and be an optimal solution for every AS. Some of them provide strong security but are lacking deployment flexibility or require more resources to run. Others are lightweight, easy to deploy, but not tackling all of the hijacking attacks.

In this research, a new way of overcoming BGP security and deployment issues was introduced. The solution is designed to use Quorum blockchain platform to exchange packets in a peer to peer way and protect the integrity of the information using modern cryptographic algorithms. Also, it uses smart contracts to store the information required by other entities to verify the received BGP messages. The security and deployment properties of such an approach were analyzed similarly to other BGP securing solutions. The new solution has demonstrated quite decent security and deployment properties which tackle all of the defined issues. The performance of such an approach was analyzed according to the BGP traffic recorded in recent years and proved to be able to handle such a load.

In the future, the solution could be improved in multiple ways. The Quorum platform has to store neighboring peers not in a local file, but in smart contracts. It is planned to be done in near future and will make the deployment much easier. Also, the smart contracts could be extended with more properties which will add more flexibility for ASes to store other routing information, e.g., network policies. Lastly, the more sophisticated testing must be performed to reveal corner cases and investigate the performance close to the real-life scenarios.

Bibliography

[1] Cisco CCNA. "Sample Configuration for Authentication in OSPF". https://www.cisco.com/c/en/us/support/docs/ip/open-shortest-path-first-ospf/13697-25.html, 2005. [Online; accessed 28-September-2018].

[2] ChainTrade. "10 Advantages of Using Smart Contracts". https://medium.com/@ChainTrade/10-advantages-of-using-smart-contracts-bc29c508691a, 2017. [Online; accessed 28-September-2018].

[3] BitCoin Info Charts. "Ethereum Statistics". https://bitinfocharts.com/ethereum/, 2018. [Online; accessed 28-September-2018].

[4] A. de la Rocha and P. Papadimitratos. "Blockchain-based Public Key Infrastructure for Inter-Domain Secure Routing". *IFIP WG 11.4 Workshop on Open Problems in Network Security (IFIP iNetSec)*, 2017.

[5] Christian Decker and Roger Wattenhofer. "Information Propagation in the Bitcoin Network". *IEEE P2P 2013 Proceedings*, https://www.tik.ee.ethz.ch/file/49318d3f56c1d525aabf7fda78b23fc0/P2P2013_041.pdf, 2013. [Online; accessed 28-September-2018].

[6] Saraju P. Mohanty Chi Yang Deepak Puthal, Nisha Saroha Malik. "The Blockchain as a Decentralized Security Framework". *IEEE Consumer Electronics Magazine vol. 7*, https://www.researchgate.net/publication/323491592_The_Blockchain_as_a_Decentralized_Security_Framework_Future_Directions, 2018. [Online; accessed 28-September-2018].

[7] Internet Engineering Task Force. "Security Requirements for BGP Path Validation". https://www.rfc-editor.org/rfc/pdfrfc/rfc7353.txt.pdf, 2014. [Online; accessed 28-September-2018].

[8] Internet Engineering Task Force. "BGPsec Protocol Specification". https://tools.ietf.org/html/rfc8205, 2017. [Online; accessed 28-September-2018].

[9] T. Griffin J. Ioannidis P. Mcdaniel G. Goodell, W. Aiello. "Working Around BGP: An Incremental Approach to Improving Security and Accuracy of Interdomain Routing". *NDSS*, https://www.researchgate.net/publication/2543280_Working_Around_BGP_An_Incremental_Approach_to_Improving_Security_and_Accuracy_of_Interdomain_Routing, 2002. [Online; accessed 28-September-2018].

[10] Network Working Group. "RFC 1105". https://tools.ietf.org/html/rfc1105, 1989. [Online; accessed 28-September-2018].

[11] Network Working Group. "RFC 1164". https://tools.ietf.org/html/rfc1164, 1990. [Online; accessed 28-September-2018].

[12] Network Working Group. "RFC 4271". https://tools.ietf.org/html/rfc4271, 2006. [Online; accessed 28-September-2018].

[13] Geoff Huston. "BGP in 2017". https://blog.apnic.net/2018/01/10/bgp-in-2017, 2018. [Online; accessed 28-September-2018].

[14] Jennifer Rexford Kevin Butler, Toni R Farley. "A Survey of BGP Security Issues and Solutions". *Proceedings of the IEEE, vol. 98*, 2009.

[15] P. Papadimitratos A. Perrig M. Hollick, C. Nita-Rotaru and S. Schmid. "Toward a Taxonomy and Attacker Model for Secure Routing Protocols". *ACM SIGCOMM Computer Communication Review, vol. 47, no. 1, pp. 43-48*, 2017.

[16] Kieren McCarthy. "OK, this time it's for real: The last available IPv4 address block has gone". https://www.theregister.co.uk/2018/04/18/last_ipv4_address/, 2018. [Online; accessed 28-September-2018].

[17] J.P. Morgan. "Quorum Readme". https://github.com/jpmorganchase/quorum/blob/master/README.md, 2016. [Online; accessed 28-September-2018].

[18] Quorum. "Quorum minting frequency". https://github.com/jpmorganchase/quorum/blob/master/raft/doc.md#minting-frequency, 2016. [Online; accessed 28-September-2018].

[19] Stephen Wolthusen Rostom Zouaghi. "A Comparison between S-BGP and soBGP in tackling security vulnerabilities in the Border Gateway Protocol". *Royal Holloway, University of London*, https://cdn.ttgtmedia.com/searchSecurityUK/downloads/RHUL_Z_final.pdf, 2004. [Online; accessed 28-September-2018].

[20] BBN Technologies Stephen T. Kent. "Securing the Border Gateway Protocol". https://www.cisco.com/c/en/us/about/press/internet-protocol-journal/back-issues/table-contents-25/securing-bgp-s-bgp.html, 2004. [Online; accessed 28-September-2018].

[21] Russ White. "Securing BGP Through Secure Origin BGP". *International Journal of Internet Protocol Technology*, https://www.cisco.com/c/en/us/about/press/internet-protocol-journal/back-issues/table-contents-25/securing-bgp-sobgp.html, 2003. [Online; accessed 28-September-2018].

Appendices

Appendix A

Implementation code

https://gits-15.sys.kth.se/tsumak/master-thesis

Appendix B

Demonstration

https://www.dropbox.com/s/t1o4hzptw0br4a1/bgp-demo.mov?dl=0