Securing BGP Communities

Design of a new RPKI object to mitigate BGP Community Attacks

R.F. Huisman

Delft University of Technology

TU Delft

Securing BGP Communities

Design of a new RPKI object to mitigate BGP Community Attacks

by

R.F. Huisman

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday July 15, 2022 at 09:00.

Student number:	4657993	
Project duration:	November 22, 2021 – July 15, 2022	
Thesis committee:	Prof. dr. ir. G. Smaragdakis,	TU Delft, Responsible Full Professor
	Dr. ir. A. Zarras,	TU Delft, Daily Supervisor
	Dr. ir. G. Iosifidis,	TU Delft, Third Committee Member

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Abstract

Research has shown that the Border Gateway Protocol (BGP) is vulnerable to a new attack that exploits the community attribute. These community attacks can influence BGP routing in unintended ways. Currently, there are no effective mitigations against these attacks which do not limit the normal usage of BGP communities or offer cryptographic authorization of community values. In this thesis, we are the first to design and evaluate a new Resource Public Key Infrastructure (RPKI) object that provides these properties, called a Route Community Authorization (RCA). Autonomous Systems (ASes) can sign and upload an RCA that includes which communities are authorized to be used and under which conditions to reduce the attack surface. The approach does not alter BGP itself and ASes that deploy the security solution can verify if route announcements with community values are allowed to use those communities. With the RPKI construction of our solution, cryptographic operations are offloaded from the router to a separate server. This is because most BGP routers deployed on the internet are not capable yet of performing cryptographic operations on every announcement.

We simulate attack scenarios and evaluate the developed solution to see if it is a viable option to be deployed on the internet. The results show that the developed solution can block the community attacks, on the condition that ASes have properly configured the RCA and at least the target, or an AS between the attacker and the target, deploys the solution. If ASes configure the RCAs too broad, then there is still room for an attacker to abuse the community values. We show that the performance of the solution using properly configured RCAs can keep up with the rate of announcements seen on the internet. Since experiments were performed in simulations, additional experiments are needed in the future on hardware routers and by ASes on the internet to see if the results are the same. The biggest issue for the developed solution is the initial adoption by ASes since security benefits are low in the beginning, but increase once more ASes participate.

Once deployed BGP routers can handle cryptographic operations on every announcement, we suggest using BGPsec and extending it to include the community attribute. Using a BGPsec approach will improve security and provide integrity of the community attribute, which RCA does not provide.

Preface

”Securing BGP Communities, Design of a new RPKI object to mitigate BGP Community Attacks” is my thesis to obtain the degree of Master of Science in Computer Science at the Delft University of Technology, with a specialization in Cyber Security. I started in November 2021 and finished in July 2022 with researching and writing this thesis.

This research topic was suggested by the PhD student Bart Hermans. The topic was challenging and fun, especially since mitigating BGP community attacks with cryptographic authorization was an area that was still unexplored at the time of writing. I want to thank Bart Hermans, Apostolis Zarras, and Georgios Smaragdakis for their excellent guidance during this thesis. Their feedback on the thesis was highly appreciated and helpful, and without their cooperation I would not have been able to deliver this result.

Finally, I want to thank my friends and family for their continued support throughout the thesis and my overall studies so far. I am proud of the final results, the thesis itself, and the overall work that I put in. I have developed a passion for this area of research during the project and for future research in this area that I hope to convey to you as a reader. I hope you enjoy reading this thesis and can either benefit from it or be inspired for future research work in the area.

Thanks!

R.F. Huisman
Delft, July 2022

Contents

1 Introduction

The internet consists of sets of interconnected networks, called Autonomous Systems (ASes). Each AS contains hosts and routers that are under the administrative control of a single entity [1]. ASes assign IP addresses to their hosts and routers and are responsible for forwarding traffic to and from their IP addresses. For ASes to communicate which routes can be used to reach an IP address, they use the Border Gateway Protocol (BGP). BGP is used to communicate network reachability information to other BGP routers on the internet. BGP plays a critical role on the internet to route internet traffic, but it provides no security guarantees [2, 3].

A large number of security proposals have been developed to make BGP more secure, but not many are adopted or are viable to deploy [3, 4]. With the few that are currently deployed on the internet, BGP still remains vulnerable to different attacks [3]. Attackers are also aware of the vulnerabilities in BGP, which they have exploited in the past. To name a few examples of attacks and manual mistakes:

- Pakistan Telecom brought down YouTube globally for two hours in 2008 in an attempt to block YouTube in their country [5].
- A BGP hijack attack redirected users of MyEtherWallet (a website hosting Ethereum cryptocurrency wallets) to a malicious clone, stealing the credentials and in turn, emptying the users' Ethereum wallets in 2018 [6].
- A large portion of European mobile traffic was temporarily routed through China Telecom in 2019 due to a route leak at a Swiss data center named Safe Host [7].

These are just a few of the attacks and misconfigurations that have happened. Research shows that many attacks are possible and what attackers can gain from it [3]. In 2018, a new attack vector was discovered against BGP, using the community attribute [8]. BGP communities are an extension to BGP that can be used to signal semantics between BGP routers within and between ASes. Streibelt *et al.* have shown that communities are widely used and that they can be exploited to influence routing in unintended ways. This is because BGP communities have complex policies, error-prone configurations, a lack of cryptographic integrity and authentication, and they propagate far beyond their intended target [8].

In 2018, the first attack was observed making use of BGP communities. In this attack, a BGP hijack targeting US payment processing companies used BGP communities to increase the propagation of the hijacked prefixes to a larger set of peers [9]. In 2019, the authors of [10] showed how adversaries can launch targeted interception attacks using BGP communities. Although research has shown the exploitability of BGP communities [8, 10, 11], there has only been one theoretical research paper yet that protects the community attribute with cryptographic authentication or integrity. Furthermore, none of the cryptographic security proposals that currently exist for BGP cover the community attribute [8]. This means that even if BGP security proposals are widely deployed, attacks enabled by BGP communities are still possible. The only extension to date for protecting communities is proposed in [11], where

the author suggests incorporating the community attributes in BGPsec. However, this proposal is only theoretical and has no concrete metrics showing that it works in practice or what the impact would be on BGP performance. Since BGPsec has many issues hindering its adoption and effectiveness in partial deployment [3], more research is needed to develop robust solutions that protect against BGP community attacks.

This thesis will try to fill that gap by being the first to develop and evaluate a solution that provides cryptographic authorization for BGP communities. The solution achieves this by creating a new RPKI[1] object where an AS can sign which communities are allowed to be used and provide several conditions that need to hold to reduce the attack surface. ASes that deploy the solution can verify if route announcements carrying communities are allowed to use that community. Verification failures can result in dropping the announcement and can thus block attack attempts.

Therefore, the main contributions of this thesis are the design of a protocol to secure BGP communities via cryptographic authorization. Such a protocol that protects community values using a cryptographic approach, while still allowing the legitimate use of communities and having adequate performance, does not exist yet. Furthermore, we do not only provide a theoretical protocol design but also implement it in code and evaluate the protocol in simulation software. The evaluation is based on metrics that are considered important for BGP security proposals and on the overall effectiveness against community attacks. The main research question that this thesis will answer is:

”How can we secure BGP community attributes against attacks?”

The thesis is structured as follows. In Chapter 2, we will discuss the relevant background for BGP, the community attribute, notable security extensions, and community attacks that is required to understand this thesis. Readers can skip the sections in this chapter that they are already familiar with. After the background, we will discuss the related work in Chapter 3. Afterward, Chapter 4 discusses the methodology of this thesis, and in Chapter 5 the design of the security solution is explained and documented. The evaluation of the effectiveness and performance of the solution is discussed in Chapter 6. Finally, we end this thesis with a discussion of the results and limitations in Chapter 7 and the conclusion in Chapter 8.

[1] Resource Public Key Infrastructure, designed to protect against prefix hijacking [12]. RPKI is explained in Chapter 2.

2 Background

In this section, we will discuss the necessary background of BGP that is needed for this thesis. To accomplish this, we first discuss the workings of BGP itself, with the community attribute in particular, and take a look at the security issues and attacks on the plain protocol. In addition, we discuss the properties that a viable security solution should have. After that, we take a look at the vulnerabilities of the community attribute, the attacks that are possible with it, and why it remains an issue to this day. We finish this chapter by discussing the most notable security extensions of BGP.

2.1. BGP

The internet consists of large sets of networks, called autonomous systems. Each autonomous system is identified by a unique AS number (ASN). The Internet Assigned Numbers Authority (IANA) is responsible for the ASNs used for routing internet traffic. IANA delegates the allocation and assignment of ASNs to Regional Internet Registries, which allocate and assign the ASNs to network operators according to their policies [13]. An AS consists of a set of routers under a single technical administration [1]. The ASes are granted a set of delegated IP addresses, which they assign to their hosts and routers. To route packets between these routers within an AS, they use an intra-AS routing protocol like OSPF[1]. To also be able to route packets with other ASes, an inter-AS routing protocol is needed. BGP is such an inter-AS routing protocol.

BGP consists of BGP routers to exchange network layer reachability information to other BGP routers [1]. Two ASes that exchange routing information are called peers. A peer in a different AS is called an external peer, and a peer within the same AS is an internal peer. Each AS has a routing policy that defines which routes to accept and which routes to publish to other ASes. BGP uses TCP as its transport protocol.

Routes on the internet are announced via BGP UPDATE messages. Each update message contains a range of IP addresses, indicated by a prefix. For example, a prefix of 192.168.0.0/24 means the IP range of 192.168.0.0 - 192.168.0.255, and 192.168.0.0/16 means 192.168.0.0 - 192.168.255.255. Each UPDATE message contains a few attributes that can fall in four categories, seen below [1]:

1. Well-known mandatory.
2. Well-known discretionary.
3. Optional transitive.
4. Optional non-transitive.

All well-known attributes must be recognized by all different BGP implementations. The mandatory attributes must be included in each UPDATE message containing IP ranges, while the discretionary

[1] https://datatracker.ietf.org/doc/html/rfc2328

attributes may or may not be included depending on the type of UPDATE message [1]. Optional attributes, on the other hand, are not required to be recognized by all BGP implementations. If they are transitive, then a BGP implementation must accept the UPDATE message and can be passed along to other peers. If the attribute is included when passing it along, then it must stay transitive (it cannot be changed back to non-transitive). If an attribute is non-transitive, then BGP implementations must ignore the attribute and must not pass this attribute along to other peers [1]. It is good to know that optional transitive attributes can be attached to an update by any routers on the path, not only the originator. For non-transitive attributes, it depends. However, both types of optional attributes can be altered by BGP implementations on the path [1]. An update message, to name a few attribute examples, contains the AS originating the prefix called ORIGIN, a sequence of ASes that have forwarded this announcement so far called AS-PATH, and the IP address of the next router called NEXT_HOP [1].

When an UPDATE message has been sent by a BGP router, the receiving BGP speaker stores the update message locally in its Routing Information Base (RIB). Routes learned from other BGP speakers are stored in the Adj-RIB-In. Based on its policy, which can be influenced by economical factors, business decisions, etc., the BGP speaker selects a set of routes to use itself. These routes are stored in the Loc-RIB, which is after the policies have been applied to the routes from the Adj-RIB-In. Finally, a BGP speaker chooses a set of routes to advertise to its peers, which is located in Adj-RIB-Out [1]. The route decision process is determined by the best path selection algorithm. This algorithm determines which route is used to forward traffic to when multiple routes exist. Multiple attributes of BGP and information about the route are used in this algorithm, but an important one is that higher preference is given to routes with more specific IP prefixes, known as the longest prefix match rule. As an example, 192.168.0.1/32 is preferred over 192.168.0.0/24 as the /32 prefix is more specific. Another important rule is the shortest AS-PATH, meaning that shorter AS paths are preferred over longer paths. As an example, if AS A receives two routes p with AS-PATH of size two and three respectively from two distinct ASes, then the route with AS-PATH size 2 is preferred. Note that business decisions and economics also influence routing decisions, as certain paths can be more expensive to use than others or a certain path has less bandwidth than another one for example. A complete overview of a best path selection algorithm used in practice can be found here for Cisco systems[2] and here for Juniper systems[3], two major hardware vendors in the BGP field.

Three types of ASes can be classified [3]. First, there are stub ASes, which only carry traffic where the source or destination is located within the AS. Second, there are stub ASes that have connections to multiple ASes, which are called multi-homed ASes. Lastly, there are provider ASes that can also forward traffic whose source or destination is not located in that AS. A different name for these ASes is transit ASes.

On the other hand, there are also Internet eXchange Points (IXPs), which provide a data interconnection facility where hundreds of ASes meet and exchange inter-domain traffic [14]. By connecting to an IXP, ASes can shorten the paths to the transit coming from other ASes, which reduces latency, round-trip time and reduces costs by avoiding routing network through providers [14].

2.1.1. Community Attribute

As we have seen above, BGP has several attributes, which can provide support in the decision making of routing policies. One of these attributes is called BGP communities. BGP communities are an optional transitive BGP attribute that are tagged to route advertisements [15].

Network operators can use BGP communities to take different actions based on the received community tags. This means that communities can be used as a common label for groups of prefixes, but also to signal semantics between ASes or within the same AS [8]. They now encode a wide variety of information and are increasing in popularity. A few examples are encoding geolocation, traffic engineering, encoding latency information, and Remotely Triggered Blackholing (RTBH) services [8]. RTBH is a service that can be offered by an AS for their customers to mitigate Denial-Of-Service attacks

[2] https://www.cisco.com/c/en/us/support/docs/ip/border-gateway-protocol-bgp/13753-25.html
[3] https://www.juniper.net/documentation/en_US/junos/topics/reference/general/routing-protocols-address-representation.html

by dropping traffic to the announced prefix [16]. Later in this chapter, we will see how the community attribute can be exploited by an adversary.

The community attribute is a 32-bit field and can take any value. However, [17] introduced large communities which takes 96-bits to accommodate larger ASN values. In this thesis, we will assume the use of 32-bit community values, but the translation to 96-bits should be trivial. The convention is to represent the ASN of the AS defining the community as the first 16 bits, and the last 16 bits as the actual 'value'. This value can either signal an action, like RTBH, or a label, like geolocation [8]. As an example, the format for a community value 567 defined by AS 1234 will look like 1234:567.

[15] defines a few standardized community labels, but this is small compared to the actual range of values used in practice [8]. These values can differ in meaning per AS, and there is no documentation for each AS as to what values they support or use, and this documentation is sometimes incomplete. There is also no policy for how ASes should handle routes tagged with communities. Each AS is free to define and use its actions and labels for the possible community values. In addition, each AS is free to add, delete and modify the communities of announcements it receives [8]. As we will see later, this freedom of community values is problematic, as well as how far community values propagate in the network [18]. Furthermore, the security extensions of BGP which we will discuss later in this section, do not protect the community attribute.

Communities have been identified to be used both internally in an AS when receiving a new route, and externally to signal an action or information about a route. [19] identified that ASes are effectively a combination of taggers or silent (they add community values or not) and cleaner or forward (remove tags or ignore). According to [19], the communities that are sent to other ASes fall in one of these categories:

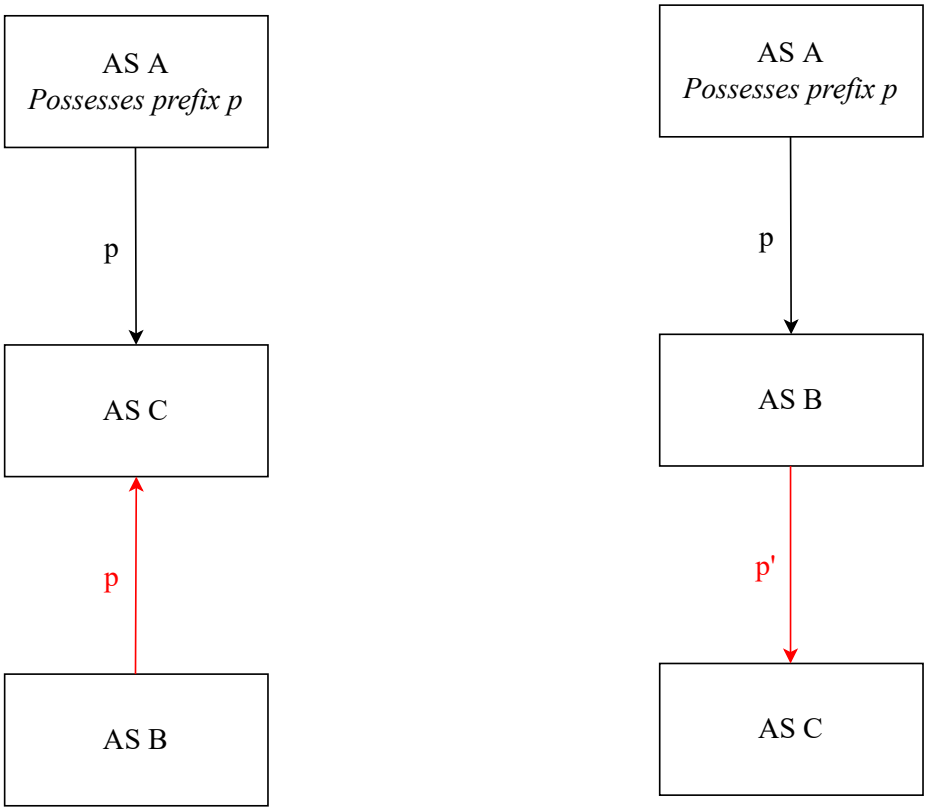
- route selection: performed by adjusting the local preference attribute (used to manipulate the best outbound path for an AS, it is set on received routes) and with AS path prepending (used to manipulate routing decisions by inflating the AS path).
- selective announcements: done by labelling routes according to which class of AS (or even which specific AS) they should be announced to.
- route suppression: performed by stating which AS not to announce to.
- blackholing: RTBH, typically indicated only for /32 prefixes to be as specific as possible.
- location: used to signal where a route has been learned, can be used for route selection.

2.2. BGP Security

Now that we have discussed BGP briefly, we will focus on the security issues with the protocol and take a look at the approaches that have been developed so far to fix them. The most important shortcoming of BGP routing is the lack of authentication and integrity [2]. Without authentication, there is no certainty that an announcement that you received is coming from the AS that it says. See Figure 2.1a for an example attack exploiting BGP without authentication. A different example showing how BGP can be exploited without integrity is shown in Figure 2.1b. In this section, we will discuss the most notable security extensions of BGP by first discussing a classification of security of BGP relevant for this thesis, before discussing a few of the security extensions. For a complete overview of BGP security, proposals, history and its challenges, the reader is directed to [3, 4].

2.2.1. Security Classification

Since we are only considering malicious use of the BGP protocol itself in this thesis, and not tampering with lower layer protocols or physical tampering with hardware or cables between routers, there are a few classifications of security for BGP [3]. First, there is securing the control plane, which means protection against tampering with routing information. Second, there is securing the data plane, meaning guaranteeing that the routers forward the packets according to the route announcements they made. In this thesis, we are only interested in securing a part of the control plane security, namely the community



(a) A is connected to C, as well as B. AS A possesses prefix p and announces it to C. B, the attacker, announces p as well. Depending on the route selection of AS C, it might choose to route traffic for p to AS B, since C has no way to check if B is allowed to originate p.

(b) A is connected to B, and B is connected to C. AS A possesses prefix p and announces it to B. B, the attacker, propagates the announcement p to C, but changes the contents (for example, changing the AS-PATH, the prefix, etc.). C receives the malicious changed announcement, but has no way to check if B changed the original announcement from A.

Figure 2.1: Two examples exploiting the lack of authentication and integrity in BGP using ASes A, B, and C.

attribute.

Next to the proactive security of securing the control and data plane, there are also reactive security measures for BGP. This includes anomaly detection and mitigation. For example, this can include monitoring services that monitor the route announcements of BGP, like BGPmon[4], and raising an alarm when an attack is detected. Another research direction is the mitigation of these attacks, which could include an automatic response that mitigates a detected attack. These are interesting and important security directions for BGP but are not the focus of this thesis. Next, we will discuss the attack classification of attacks on the control plane.

2.2.2. Attack Classification

According to [3], the following attack taxonomy can be identified for the control plane, starting with data falsification:

- Prefix hijack: An AS announces a prefix that it is not allowed to originate.
- Subprefix hijack: An AS announces a sub prefix (a subset of a prefix) that does not belong to the AS. Since sub prefixes are more specific than the original prefix, there is a high chance that, if the attack succeeds, other ASes will adopt the route.
- AS-PATH forgery: An AS changes the AS-PATH of a route announcement. This can either create a one-hop prefix hijack by announcing a fake link between the AS and the victim or announce

[4] "BGPmon monitors the routing of your prefixes and alerts you in case of an 'interesting' path change": `https://www.bgpmon.net/`

a fake link to the sub prefix of the victim (which is called a one-hop sub prefix hijack). Another variant of AS-PATH forgery is the AS-PATH prepending that we have seen before, or removing ASes in the path to create a shorter, more attractive route.

- Interception attack: An AS uses previously mentioned attacks to redirect traffic through himself (for eavesdropping purposes for example, or setting up a Man-In-The-Middle (MITM) attack) and forwards it back to the legitimate AS. In this attack, connectivity is not affected if done correctly, but victims will see an increase in latency.
- Replay/Suppression attack: An AS replays or suppresses an older route announcement.
- Collusion attack: Two ASes that are not direct peers of each other create a BGP session with each other through a virtual tunnel. Through this virtual link, they can forge routes without causing suspicious routing conflicts.

Next to data falsification attacks, there also exist protocol manipulation attacks where an AS manipulates properties of the BGP protocol. Lastly, there also exist data misuse attacks, where an AS uses correct routing data for malicious purposes. This can cause a Denial of Service (DoS) or a route leak. A route leak happens when an AS announces routes to peers that are not intended to receive said routes [3]. For example, business contracts might prohibit announcing those routes. Route leaks can cause unintended routes that slow the internet down or route unintended traffic through a small AS (due to a provider leaking a route) that is not able to handle the load.

2.2.3. Desired solution properties

As stated in [3, 4, 20, 21], securing BGP is a difficult task. There are many factors to consider, and many proposals have failed to get them right. Since we want to develop a new extension for BGP in this thesis, it is important to consider these properties and make the right balance between them. Since there is no free lunch in computer science, there is no such thing as perfect security and performance, so sacrifices need to be made. The compromises and balancing of our proposal is documented in Chapter 5. Based on [3], the following properties for security solutions can be identified:

Security properties

How much extra security does the new extension guarantee? Which vulnerabilities remain for BGP when deployed? Furthermore, new proposals should not introduce new attack vectors that do not exist in legacy BGP.

Performance properties

- Convergence delay: security always comes at a performance cost, either in computational overhead or additional communication costs. This performance overhead can cause a delay for routers to reach a stable state, where they settled on the best route. The time it takes to reach this state is the convergence delay and is compared to the convergence delay of plain BGP without the security extension deployed.
- Stability: a stable state is important in BGP since the quality of connections is otherwise decreased or even unusable. Stability is measured if convergence on the best route is ever reached between routers. If due to the new security extension the stability is compromised, it is unusable for deployment.
- Scalability: due to the additional overhead of the security extension, the complexity of the protocol is increased. Scalability measures how well the new protocol scales as the number of ASes increases. It should scale when fully deployed on the internet, otherwise, it cannot be deployed.
- Computational overhead: As discussed before, security increases computational and communication overhead. The computational overhead can be defined as how many BGP control messages can be processed per unit of time, or as how many operations need to be executed on each message compared to legacy BGP. This needs to be measured on the same hardware, but a compromise can be made to install better hardware depending on the additional security guarantees a protocol can guarantee (but this new hardware needs to be justified).

- Bandwidth overhead: Additional communication overhead impacts the bandwidth needed for BGP to function. More overhead can be costly, both in terms of the rate of BGP messages to be sent, but also economically depending on how much the bandwidth costs.
- Storage overhead: New security protocols can require more memory on the BGP routers to perform the necessary computations compared to legacy BGP. The degree of additional memory required is called the storage overhead and also includes the case if additional hardware is required to deploy the security extension.

Privacy properties
Certain ASes want to keep their routing policies, business contracts/relationships, or other data private. A new protocol should not require additional sensitive information that is revealed compared to plain BGP, as ASes will otherwise not want to deploy it.

Deployability properties
The above three properties are important to consider for BGP, but they do not consider the deployability of the solution. A new solution should ideally be easy to deploy incrementally on the internet and also provide immediate security benefits. If only security benefits are guaranteed when all ASes deploy the new protocol, or everyone needs to deploy it at once, then the protocol will never be deployed in the real world. A large number of ASes exist (73444 at the moment of writing [22]) and they all have their own hardware. Certain ASes might not care or have the money/time to deploy a new protocol version right away, and the incentives for an AS to deploy the security proposal are minimal if they do not see fast security improvements. In addition, interoperability with legacy BGP is needed because not all ASes can deploy a new version right away, or want to deploy it.

2.3. Community Attribute Vulnerabilities

The BGP community attribute introduces several new attack vectors due to a lack of integrity and authenticity, the complex policies of ASes, error-prone configurations, missing semantics, and the wide degree of community propagation in the network [8]. Streibelt *et al.* found that due to the rich connectivity of transit ASes, received BGP communities are effectively propagated globally. In this section, we will look at why community attacks are possible, which attacks can be performed, and how they can be (partly) mitigated.

2.3.1. Community Attacks

Three possible attacks can be classified as identified in [8]. There are multiple scenarios under which these types of attacks can be performed, but only one will be given to explain the concept:

- RTBH: As discussed before, the RTBH community value drops traffic to the destination under attack, and can be added to a route announcement that should be as specific as possible. Note, that this should also be done as close to the source of the attack as possible. However, Streibelt *et al.* identified 50% travelled up to two hops [8]. This service offered by ASes should only be used when an IP address is under attack and the announcer should have authority over the IP address. Due to a lack of authenticity, this allows for an RTBH attack. See Figure 2.2 for an example.
- Traffic steering: An easy example to illustrate traffic steering is with the community value that enables path prepending for ASes that define that service. If an adversary performs path prepending, it can make a route less attractive for the AS that receives the altered announcement. Due to this, the AS might choose a different path to route traffic.
- Route manipulation: Some IXPs offer specific services that are enabled by community values, and one of those services is signalling which routes should be advertised to other IXP members or not. An attacker could use this service to cause the IXP to not advertise certain prefixes to peers. This is just an example of route manipulation, and this example is possible since some IXPs handle community-based services in a specific order. In this case, the 'do not advertise' community is handled before the 'do advertise' community that was sent by the legitimate AS [8].

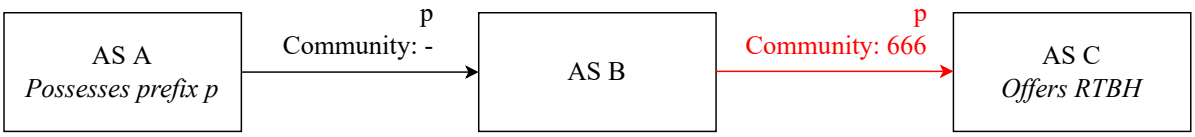


Figure 2.2: RTBH attack by AS B. AS C offers RTBH service and B adds the blackhole community (value 666) to the route announcement of A before forwarding it.

Streibelt *et al.* identified two conditions that are necessary or sufficient to perform community attacks:

1. Necessary condition: Remotely triggered actions based on community values can be used if communities are propagated beyond a single AS and the community service is known (or guessed/inferred by the attacker).

2. Sufficient condition: For the community vulnerabilities to be triggered, it is sufficient if the attacker can advertise prefixes with the appropriate communities for all ASes on the path to the target AS.

The authors confirmed that necessary conditions hold on the internet due to communities propagating beyond their direct neighbours [8]. They also carried out the attacks in the wild, and provide a summary of their insights, which can be seen in Figure 2.3.

Scenario	Hijack	Insights gained from running experiments in the wild	Difficulty
Blackholing	no	Allowed prefix length is checked; activation of RTBH service is typically required.	easy
	yes	Allowed prefix length is checked; origin validation was not always checked, thus the attack was easier.	easy
Traffic Steering with local pref	no	The business relationship of the attacker with the attackee or transit networks is checked – the flattening of the Internet makes this attacks hard to launch (providers only act on communities set by their customers).	hard
	yes	The business relationship of the attacker with the attackee or transit networks is checked – the flattening of the Internet makes this attacks hard to launch (providers only act on communities set by their customers); IRR records for origin validation are typically checked, but the check can be circumvented.	hard
Traffic Steering with path prepending	no	The business relationship of the attacker with the attackee or transit networks is typically checked – the flattening of the Internet makes this attacks hard to launch (providers only act on communities set by their customers); AS path prepending has typically low evaluation order, thus the attack may not be successful.	hard
	yes	The business relationship of the attacker with the attackee or transit networks is typically checked – the flattening of the Internet makes this attacks hard to launch (providers only act on communities set by their customers); IRR records for origin validation are typically checked, but the check can be circumvented; AS path prepending has typically low evaluation order, thus the attack may not be successful.	hard
Route Manipulation	no	Requires inference of community evaluation order when this information is not public.	medium
	yes	Requires inference of community evaluation order when this information is not public; IRR records for origin validation are typically checked, but the check can be circumvented.	medium

Figure 2.3: Summary of insights from attacks carried out in the wild by Streibelt *et al.* in [8].

2.3.2. Possible Mitigations

The authors of [8] identified with network operators that the scope of community values is usually only one or two hops, but based on observations, the communities propagate beyond that. Streibelt *et al.* came up with a few mitigations. One mitigation is deciding if the benefits of easy signalling using communities is worth the risks of potential attacks. The second mitigation is filtering the communities to limit the reach. The third is the call for authentication of the right to attach a community to an announcement or modify one in transit, but the authors list that there are no known means to do this. Lastly, monitoring more globally could discourage malicious parties from carrying out attacks, since there is a high risk of being caught. However. there is no global BGP view and route collectors do not see every announcement or community as some ASes filter or change them [8]. Since there is no clear semantics for the communities, these 'undefined' values cannot be monitored properly for what the consequence might be. That is why Streibelt *et al.* call for proper documentation. Community values still pose a clear threat and there exist no completely effective mitigations yet.

Birge-Lee *et al.* found that you can also perform interception attacks using BGP communities. However, from the currently available mitigations, only prefix filtering could stop the attack. Route origin validation and AS-PATH filtering are ineffective against an interception attack [10]. The authors list a few possible countermeasures, namely restricting community propagation, restricting community size in announcements, restricting community actions, and anomaly detection based on historical updates. All of the countermeasures they list can stop the interception attack, but limit the legitimate use of BGP communities [10]. The authors do note that BGPsec offers the most comprehensive resolution to BGP interception and hijack attacks, but BGPsec does not prevent all community attacks and has severe issues that hinder deployment. Both of these issues of BGPsec will be explored in the next section.

2.4. Notable BGP security extensions

There have been a vast majority of security proposals for BGP. [3, 4] made an extensive overview of the current state of BGP security. An overview of these security proposals and what they can protect against can be seen in Figure 2.4, which has been made by the authors of [3]. As can be seen, there are many proposals, but only a few are actually adopted or even standardized. A big issue with BGP security as well is that many proposals are only theoretical academic papers. In addition, many of the proposals do not meet the desired properties for security proposals that we introduced before, making them not a viable option (see Figure 2.4). For the notable security extensions, we will only consider the ones that are (partially) adopted and BGPsec. In addition, since the release of [3], there have been a few interesting proposals. One of them is ASPA, which we will also take a brief look at.

Properties	Approach															
	Route filtering	IRR	Use of DNS	IRV	RPKI	RPKI enhanced	S-BGP	soBGP	psBGP	BGPsec	S-BGP enhanced	SPV	SMPC-based routing	Blockchain-based routing	Listen & Whisper	Use of traceroute
Security																
Control-/ Data-plane	■ / □	■ / □	■ / □	■ / □	■ / □	■ / □	■ / □	■ / □	■ / □	■ / □	■ / □	■ / □	■ / □	■ / □	■ / ■	□ / ■
Attacks Covered:																
Prefix / Subprefix hijack	◪ / ◪	■ / ■	■ / ■	■ / ■	■ / ■	■ / ■	■ / ■	■ / ■	■ / ■	■ / ■	■ / ■	■ / ■	■ / ■	■ / ■	■ / ■	□ / □
AS path forgery	◪	■	□	■	□	◪	■	■	◪	■	■	◪	■	■	□	□
Interception attack	◪	■	□	■	□	◪	■	■	◪	■	■	◪	■	■	◪	□
Replay/Suppression attack	□	□	□	□	□	□	□	□	□	◪	□	□	■	■	□	□
Collusion attack	□	□	□	□	□	□	□	□	□	◪	□	□	□	□	□	□
MED modification	□	□	□	■	□	□	■	■	□	□	□	□	■	□	□	□
Exploit RFD/MRAI timer	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□
Denial of service	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□
Route leak	◪	■	□	■	□	□	□	□	□	◪	□	□	■	■	□	□
Performance																
Convergence delay	□	□	□	■	□	□	■	■	■	■	■	◪	■	■	◪	■
Stability	■	■	■	◪	■	■	◪	◪	◪	◪	◪	◪	◪	◪	◪	◪
Scalability	◪	◪	◪	□	◪	◪	□	□	□	□	□	◪	□	□	□	□
Computational overhead	□	□	□	◪	□	□	■	■	■	■	◪	◪	■	■	■	■
Bandwidth overhead	□	□	□	■	□	□	■	■	■	■	◪	◪	■	■	■	■
Storage overhead	■	■	■	■	■	■	■	■	■	■	■	◪	□	■	■	■
Privacy																
Routing privacy	◪	□	□	◪	◪	◪	◪	□	◪	◪	◪	◪	■	■	◪	□
Deployability																
Deployability	■	■	■	□	■	■	■	■	■	■	■	■	□	□	■	■
Adoptability	◪	◪	□	□	◪	◪	□	□	□	□	□	□	□	□	□	□
Status																
Adopted / Standardized	◪ / □	◪ / □	□ / □	□ / □	◪ / ■	□ / □	□ / □	□ / ◪	□ / □	□ / ◪	□ / □	□ / □	□ / □	□ / □	□ / □	◪ / □
Academic paper only	□	□	■	■	□	■	■	■	■	□	■	■	■	■	■	◪

Figure 2.4: A summary of main BGP security proposals from [3]. Notion: ■ has feature, ◪ partially has feature, □ does not have feature.

Route filtering

ASes can use route filtering to filter out bogus route announcements, caused either due to misconfigurations or malicious attempts. A few examples of filters are filtering out IP prefixes (like private or special IP ranges), a match on AS paths, or any other attribute of a BGP prefix [23]. But network operators can also use a publicly available database containing internet routing information from other ASes [23]. Such a database is the Internet Routing Registry (IRR). The IRR provides a global view of the routing information where ASes can upload their routing data. This can be used to filter out announcements that do not comply with the data of the IRR for example. However, the IRR is flawed due to complex routing policies, the distributed nature of the database, security issues, and the accuracy of the database [3]. For example, some ASes do not share their routing information due to privacy issues (due to a lack of access control as to who can see which information). Furthermore, due to a lack of

authentication, ASes could advertise a prefix not belonging to them in the IRR and create blackholes and other security threats [23].

Peerlocking
Peerlocking was presented in 2016 as a deployable solution to defend against route leaks. ASes communicate out-of-band (meaning not via BGP itself, but via a different channel) a protected AS and a protector AS [24]. The protected AS communicates designated authorized upstreams to the protector. The protector then filters out route announcements whose AS-PATH contains the protected AS unless it is received directly from the protected AS (or one of its designated upstreams, if peer locking is deployed at a lower-tier AS). A requirement for this to work is trust between the ASes that are peerlocking. The filter prevents the protector AS from propagating its traffic onto any leaked routes that transits the protected AS [24]. Several tier 1 ASes (the ASes do not have providers themselves) are already using peerlocking [24]. A light version that requires no communication is also deployed by networks to filter out tier 1 ASes from announcements and thus prevent tier 1 leaks. This is based on the assumption that provider ASes should never receive a route from a customer where the AS-PATH includes a tier 1 AS [24].

RPKI
The SIDR working group developed a Resource Public Key Infrastructure (RPKI) for BGP, to provide origin authentication for route announcements. RPKI provides a set of building blocks that allows various levels of protection of the routing system, with the initial goal being route origin validation [12]. This improves upon some of the issues with IRR, where authentication is not present. Each AS can upload a cryptographically signed Route Origin Authorization (ROA) in a regional database that provides cryptographic proof that that AS is the rightful owner of a prefix and is thus allowed to announce it. RPKI data is delivered via a validated cache using the RPKI-RTR protocol to the BGP routers. The router itself does not perform any cryptographic validation, this is all handled by the validated cache. This setup improves the deployability of RPKI by causing minimal overhead for BGP routers and a negligible influence on convergence speed [25]. There are three states when a route announcement is checked against a ROA. Valid means the route is covered by at least one ROA, Invalid means the announced prefix is not covered by a ROA (for example, if announced by an unauthorized AS, but this could also mean that the ROA is expired), and NotFound means that the prefix is not covered by a ROA [25]. These three states allow for incremental deployment of BGP as more ASes upload their ROAs. An issue with this approach is maintaining an up-to-date view of the regional databases, since stale records and other misconfigurations have a direct impact on reachability of affected prefixes [26].

Each ROA contains the following information [27]:

- Version number.
- Timestamp.
- ROA Name.
- ASN of the origin AS that is authorized to announce the prefixes in this ROA.
- Validity date start and end, where the ROA is considered valid.
- Prefix and prefix length to speficy the range of IP addresses that are associated, and thus authorized, to be announced by the origin AS specified earlier. Multiple prefix, prefix length, and max length tupples can be specified after each other.
- Max length to specify the smallest prefix length that is allowed for this prefix to be announced. This is an optional field.

Several tools are available to easily deploy RPKI validation for routers[5] and an increasing number of ASes enforce RPKI filtering [26]. Although global adoption is still relatively low[6] at the time of writing, RPKI is currently the most widely deployed cryptographic security measure for BGP and the number

[5] https://rpki.readthedocs.io/en/latest/ops/tools.html
[6] https://rpki-monitor.antd.nist.gov/

of ROAs is increasing over time[7]. The main factors that hinder adoption are human errors causing invalidity of RPKI objects and dependencies between organizations where ASes need to wait for others to deploy RPKI first [3, 28].

One of the remaining issues is that RPKI does not provide integrity, so a lot of attacks are still possible. To further complicate the issue, RPKI itself can be circumvented with a one-hop prefix hijack [3]. As an example, see Figure 2.5. Because RPKI does not provide integrity, this type of hijack is still possible. However, it does inflates the AS-PATH compared to a normal prefix hijack, which can make the route less preferable and thus the attack less successful depending on the circumstances.

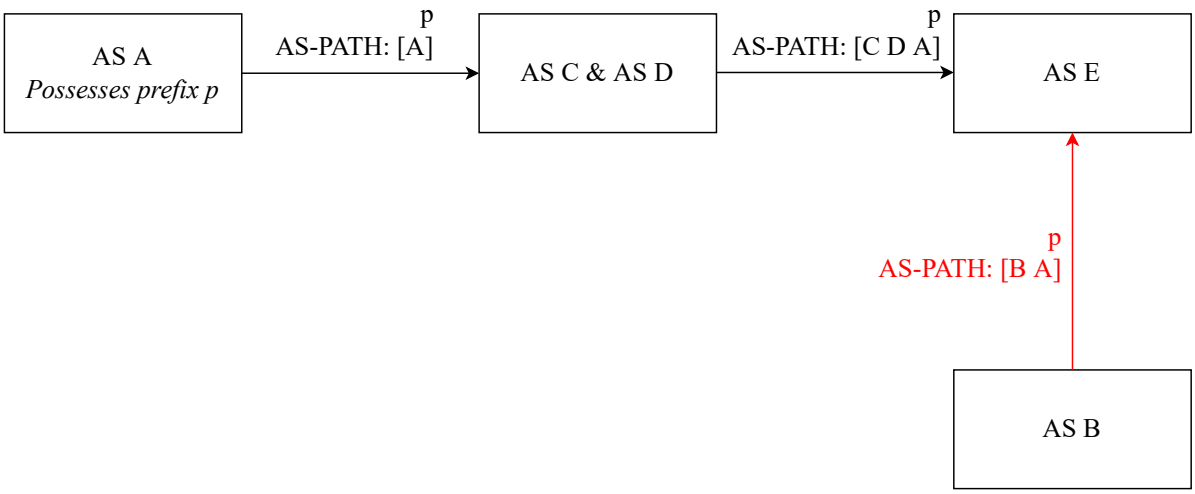


Figure 2.5: One-hop prefix hijack using ASes A, B, C, D, and E. We will assume that all ASes in this example deploy RPKI. AS A possesses prefix p and announces it via AS C and D to AS E, which makes the AS-PATH of length 3. AS B announces p as well, but it is not authorized to do so. Instead, B adjusts the AS-PATH and says it has a route to AS A which it does not have. AS E receives the announcement and cannot check if B has a path to A. Since A is authorized to announce p, E will accept the announcement of B. Since it has an AS-PATH of 2, it is likely that E will route traffic through B.

BGPsec

The lack of integrity of RPKI is where BGPsec comes into play. BGPsec provides authenticity and integrity of route announcements by also protecting the AS-PATH with authentication and integrity [29]. BGPsec uses RPKI for origin authentication. Furthermore, each router has a certificate to sign the route announcement received from the previous router. This creates a chain of cryptographic signatures that can be checked to make sure that the first AS can originate the prefix and that every router on the path did not alter the contents of the AS-PATH. This creates the following property for a path of BGPsec-enabled routers: ”every Autonomous System (AS) on the path of ASes listed in the UPDATE message has explicitly authorized the advertisement of the route to the subsequent AS in the path” [29].

The issue with BGPsec, however, is the need to replace or deploy extra hardware to support BGPsec due to the heavy computations required for signature validation [3, 4]. Furthermore, the announcements increase in size due to each signature, which has an impact on the bandwidth. A different issue with the protocol is the lack of benefits when BGPsec is deployed partially, meaning the incentive for adoption is low [3]. The biggest issue of BGPsec, however, is the ability to perform a downgrade attack [29]. BGPsec can only be used if every router on the AS-PATH supports the protocol. If this is not the case, legacy BGP is used which has all of the issues that BGPsec tries to solve. A malicious AS can thus say that it does not support BGPsec and carry out path hijacks and other attacks again.

The reader might wonder if BGPsec is able to protect against the community attacks that we have seen earlier, since it provides authenticity and integrity. Unfortunately, this is not the case. BGPsec only provides authenticity and integrity for the AS-PATH attribute [29], meaning the community attribute

[7] https://rpki.cloudflare.com/

is still unprotected [8]. Community attacks are thus still possible even when BGPsec is used.

ASPA
As we have seen, the lack of AS-PATH verification in RPKI is one of the biggest security issues that remains after deploying RPKI. BGPsec tries to solve this but fails to meet the metric requirements of a security extension that we discussed. Furthermore, the downgrade attack makes it possible to drop all BGPsec security measures, making BGPsec not suitable for any deployment soon before these issues are addressed and the necessary hardware upgrades have been made.

This is why Autonomous System Provider Authentication (ASPA) is an interesting and promising addition to RPKI. It is currently still in the draft stage at the time of writing, but the workings of ASPA and the promised security benefits are exciting. The basic idea is that if a valid route is received from a customer or peer, then there must be a customer-provider pair present in the AS-PATH [30]. Because of this, if we have a validated database of customer-provider pairs, then we can verify the routes received from customers and providers. The validation can be done by introducing a new RPKI object. The new object binds a set of provider ASNs to a customer ASN, which is signed by the customer AS [30]. ASPA is incrementally deployable and scalable and looks to be similar in computational requirements to RPKI. It brings benefits even in partial adoption and hardware additions are most likely not required to deploy ASPA as it works on the existing RPKI architecture [30].

ASPA has similarities to Peerlock, but Peerlock requires manual labour to coordinate authorizations between peers. ASPA is intended to replace Peerlock deployments according to the draft [30]. However, it is good to note that ASPA has limitations. Customer ASes need to maintain a correct set of providers in the signed objects. Furthermore, ASPA only allows the detection of misconfigurations and malicious attempts of announcements received from customers and peers [30]. There is no explicit integrity protection on the AS-PATH. This means that the upstream providers become a trusted source, meaning that these providers can hijack their customers or send malformed AS-PATHs to customers [30]. The authors do note that we have to consider the likelihood of this since providers have legal consequences for doing such practices.

2.5. BGP Alternatives

There are also alternative internet architectures that try to solve problems that plague the internet today, which is not limited to BGP. A few interesting and promising approaches are SCION (Scalability, Control, and Isolation On Next-Generation Networks)[8], RINA (Recursive InterNetwork Architecture[9], and NDN (Named Data Networking)[10]. Each of these proposed future internet architectures is solving the problems of the internet differently, and they may have problems on their own. There is much research going on in these emerging architectures and it is good to keep an eye out for them. The treatment of these internet architectures is out-of-scope for this thesis, but we have left a few footnotes for readers interested in them and to make people aware that there are other architectures than BGP in development, and possibly used in the future. Also check out 2STIC, a joint research program to ”develop and evaluate mechanisms for increasing the security, stability and transparency of internet communications, both through extensions of today’s Internet as well as through emerging internet architectures.”[11]

[8] https://scion-architecture.net/
[9] https://csr.bu.edu/rina/index.html
[10] https://named-data.net/
[11] https://www.2stic.nl/index.html

3

Related Work

In this chapter, we will discuss the related work that has been done by researchers on both BGP security and security issues of the BGP community attribute relevant for this thesis. In the background, we briefly discussed how BGP and the community attribute works, the notable security extensions to BGP and desired security properties for extensions, and discussed the taxonomy of community attacks and a few possible mitigations. We also discussed an interesting security extension called ASPA, which provides AS-PATH validation. The implementation of ASPA consists of creating a new RPKI object, which is a source of inspiration for the solution that we will develop in Chapter 5. Here, we will look at other related work in the area of BGP and community attacks. This includes attack detection, which is a reactive approach to dealing with security, contrary to the proactive approaches we have seen so far.

Many papers have talked about the security issues of BGP and provided future directions for later research, which we briefly discussed in Chapter 2. Because of the many metrics that security proposals need to keep in mind (see Section 2.2.3), and since many proposals are only academic [3], BGP security extensions are hard to develop and difficult to deploy quickly. ASes are hesitant and have a lack of incentives since many solutions are suboptimal in terms of metrics. Both the authors of [3] and [4] provide an extensive and up-to-date survey of the current state of BGP security and what has been done in the past, together with the issues and difficulty of securing BGP. Multiple papers, also covered by [3], try to look at a more practical security, including better monitoring and automatic mitigation, which is certainly an interesting direction for securing BGP. As an example, [31] provides a system for detecting serial hijackers.

Testart and Clark provide an in-depth overview of the state of routing security, adoption of security proposals, the problems with them, and directions for the future [20]. The authors also introduce a new concept called 'zone of trust', using the enhanced operational practices of MANRS participants. Their approach is compared to ASPA, with the main difference that ASPA requires a global database and that in ASPA any AS can inspect an announcement for an invalid path instead of only a MANRS participant. Due to issues of centralizing records globally in databases [20], our solution in Chapter 5 will also use include an approach without a centralized database.

Streibelt *et al.* introduce the threat of community attacks to the world and provide the first attack taxonomy in [8]. They showed why communities can be an attack vector and why more research is needed in understanding the use of communities, but also to research new mitigations for the threat. They show the difficulty of performing community attacks and how practical they are to achieve on the internet. Furthermore, Krenc *et al.* stress a second unintended side effect besides the security vulnerabilities of communities, namely an increase in unnecessary BGP routing messages due to the large propagation of community values in the network [18]. A change in community value induces update messages throughout established routes that just update the community value. In later research, Krenc *et al.* presented the first classification of BGP communities, namely the tagger, forwarder, cleaner, and silent ASes [19] that we discussed in Chapter 2.

On the other hand, Birge-Lee *et al.* show with their SICO attack that advanced interception attacks can also be achieved with communities. The attack they introduced is novel and shows how advanced interception attacks can be performed live on the internet, and that 83% of multi-homed ASes is capable of launching these attacks [10]. This only raises the importance of finding good mitigations, since we have seen in Chapter 2 that current mitigations are lacking. To further stress issues with communities, Nawrocki *et al.* talks about the issues of RTBH practices at IXPs. They reveal that only a third of RTBH events correlate with DDoS attacks, 2000 RTBH events are announced for prefixes of clients in DSL networks, that RTBH only drops 50% of traffic on average, and that some (even large ISP) operators do not even accept /32 prefixes for blackholing [32]. They also find that there is collateral damage when using RTBH since RTBH stays active for a long time after the actual attack is finished.

A first cryptographic security proposal for BGP communities was introduced by Miller and Pelsser. The authors presented a taxonomy of BGP blackholing attacks and researched that fully/partially deployed RPKI or BGPsec do protect against some of the attacks, but not against all. Furthermore, none of them protect against on-path attackers [11]. As a first step towards a cryptographic security extension for communities, the authors propose to extend the AS-PATH attribute to also include community values in the signatures. The proposal is only theoretical, so there are no concrete performance numbers. Since BGPsec is not a viable option for deployment yet (see Chapter 2), and that adding more values in the AS-PATH will only inflate the size of messages even further, this approach is not a viable option for protecting community attributes currently.

Since our security proposal builds upon RPKI, it is a good idea to understand the issues that face RPKI itself and the deployment. Gilad *et al.* talks about why RPKI has deployment issues and that wide deployment is not sufficient to protect against prefix and sub prefix hijacking. Furthermore, the authors also provide a survey with network operators to identify the causes of slow RPKI adoption [28]. This mostly consists of loss of traffic due to badly issued ROAs and inter-organization dependencies (waiting for another AS to deploy RPKI for example). Fortunately, many obstacles can be avoided according to the paper by focussing on large ISPs, making modest changes to RPKI, and using detection systems [28]. To further understand the hesitance of ASes to deploy security proposals, Sermpezis *et al.* surveyed network operators to identify the reasons that hinder the deployment of defence mechanisms in general. The importance of characteristics of a defence system, according to network operators, is summarized in Figure 3.1.

A different issue of RPKI is the importance of good ROA management. Testart *et al.* discuss the need to regularly sync local RPKI data to make sure there is no loss in traffic [26]. Hlavacek *et al.* on the other hand, try to solve the manual and error-prone certification process of RPKI issues by using automatic deployment [33]. Any error made can result in loss of traffic, so the authors came up with a novel system called DISCO, which automates certification and generates appropriate filtering rules. Furthermore, the system is tested live on the internet using the PEERING testbed[1] and is considered deployable by the authors [33]. The approach they took for implementing DISCO in a deployable way is an inspiration for our developed protocol, which we will talk more about in Chapter 5.

While a lot of research is done on new security proposals for BGP or discovering new vulnerabilities, NIST has researched an architecture to overcome the deployment issues of RPKI and BGPsec, including performance metrics [35]. In addition, to overcome the one-hop prefix hijack of RPKI, Cohen *et al.* focus on extending RPKI using path-end validation. Their approach does not require upgrading, replacing BGP routers or performing much more expensive cryptographic operations [36]. It provides added security benefits on top of RPKI by preventing the one-hop prefix hijack. Lastly, there is also an organization called MANRS[2]. It is a global initiative to provide guidelines and concrete actions for network operators to follow to reduce the most common routing threats. MANRS outlines four concrete actions for participants [37]:

[1] PEERING provides testbeds for live BGP studies, including new security proposals, to test live on the internet as a means of mitigating the obstacles that come with executing experiments on the internet. The platform supported over 40 experiments and 15 publications already in 2019 [34].

[2] Mutually Agreed Norms for Routing Security: `https://www.manrs.org/`

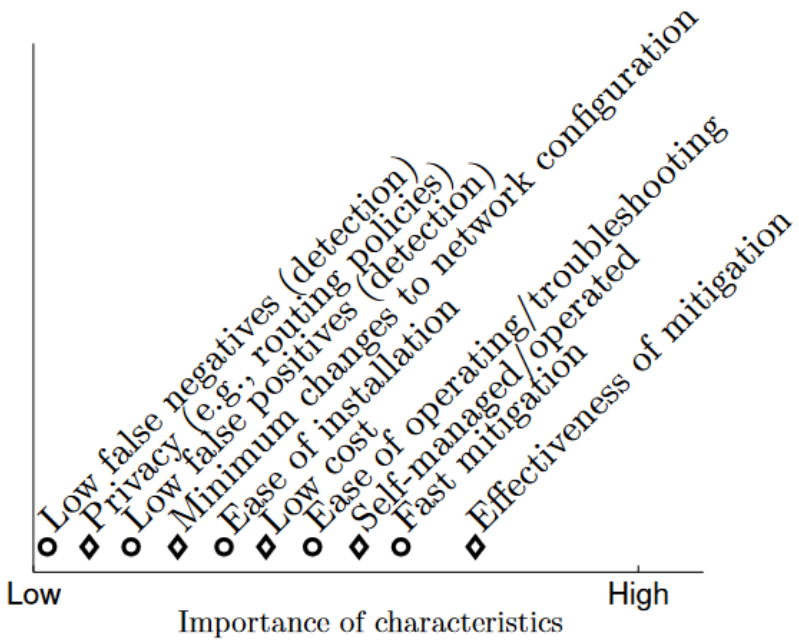


Figure 3.1: Importance of characteristics of a defense system, according to network operators [21].

- Filtering: preventing propagation of incorrect routing information.
- Anti-Spoofing: preventing traffic with spoofed source IP addresses.
- Coordination: facilitating global operational communication and coordination between network operators.
- Global Validation: facilitating validation of routing information on a global scale.

4

Methodology

In this section, we will explain and motivate the setup of this research and the approaches that will be taken. This includes what the research questions are and how they will be answered. In addition, we discuss and motivate the approaches for the data collection of the experiments needed to answer the research questions and define the scope of this thesis. The tools that are going to be used, development of the security solution in theory and practice, tools used to perform the experiments, etc. are discussed in Chapter 5 and how they impact the goal of this research and its outcomes. The experiments of this thesis are defined, explained, and discussed in Chapter 6, together with any limitations of the approaches taken. This makes it easier to understand and reference the experiments when discussing the results for the reader.

4.1. Research questions

As discussed in the introduction, the main research question for this thesis is: ”How can we secure BGP community attributes against attacks?”. To be able to answer this question effectively, we have come up with four sub-questions that will progressively help us to arrive at an answer. These sub-questions are as follows:

1. What has already been done to secure BGP against community attribute attacks?

2. What is important to consider a solution viable for deployment, and with which metrics can we measure that?

3. How can we meaningfully develop and test solutions without deploying them live on the internet?

4. How can we verify the community attribute while satisfying the requirements for a viable solution?

4.2. Approach

Now that we have defined the main research question and its sub-questions, we need to define the approach to answer each question and the tasks that need to be performed. For each sub-question, we will list the approach to answer it and any tasks that are required to be completed. An overview of the main outcomes of each subquestion will be presented in Chapter 7.

4.2.1. What has already been done to secure BGP against community attribute attacks?

This question will be answered by performing a literature review to find the current state of BGP security, the state of BGP community attacks, and the state of current mitigations against the attack. The main outcomes will be presented in Chapters 2 and 3.

4.2.2. What is important to consider a solution viable for deployment, and with which metrics can we measure that?

This question will be answered by reviewing current BGP security solutions and analyzing why they are either successful or not. Furthermore, it is important to see what the stakeholders, in this case the ASes, find important to consider a solution viable and what they are looking for. In addition, to find out important metrics, we can look at what other BGP security solutions have used to measure performance and viability for deployment. These metrics must be independent of the security solution being developed, i.e. they are not chosen or cannot only be applied to this security solution. It should be a generic approach for assessing BGP security solutions.

4.2.3. How can we meaningfully develop and test solutions without deploying them live on the internet?

This question requires us to research how we can develop BGP security proposals and test them without having access to production equipment and the rich connectivity of ASes live on the internet. The outcome of this question will be a combination of tools, software, or approaches, to develop and test BGP security solutions, and should be independent of the security solution being developed. There will be limitations by taking this approach, and these limitations must be clear and documented.

4.2.4. How can we verify the community attribute while satisfying the requirements for a viable solution?

To answer this question, we need to make a theoretical model of the proposed security solution. With this model, we can analyze what the solution can and should protect against, and we can analyze the properties of the security model using a theoretical analysis. This theoretical model can predict how the performance will be, what attacks it can defend against and which not, if the BGP protocol should be changed, etc. To arrive at and develop such a model, we can look and make use of previous security proposals for BGP and their pros and cons. Furthermore, with some creativity, we could also develop a new approach of mitigating attacks or take inspiration by taking a look at other security solutions outside of BGP. The outcome should guarantee a result where the use of community attribute values can be cryptographically authorized while keeping the metrics that we defined in the previous question in mind. Through trial and error, or incremental changes to the model, we can improve if any issues are found. Somewhere in the metrics, compromises must be made to arrive at a balanced solution. These compromises must be explained and documented.

4.3. Data collection

To answer some of the subquestions, and of course the main research question, the developed security solution needs to be evaluated. We have defined the metrics with which we can assess BGP security solutions in the background chapter. These metrics can be expressed in different ways, depending on the capabilities of the simulations and of course time constraints for this project. For this thesis, the following options are possible:

1. Theoretical results: certain metrics might not be measurable in an exact number depending on various reasons. This could be due to limitations of the simulations, lack of data, time constraints, etc. If a metric is expressed with only a theoretical analysis, this analysis should be sound and clearly explain why it is only theoretically analyzed.

2. Approximate results: for certain metrics, it is not needed to know exactly how fast the solution is. For example, speed can depend on a lot of factors, and will give different results depending on the hardware it runs on, and also if it is running on actual routers or in software only. We might only be interested in seeing if the difference between the metric of the plain solution and the proposed solution is negligible. For example, if we want to measure if the start-up time of a router is significantly impacted by the security solution deployed, we are only looking if the results are acceptable. This means that if a normal router takes 10 seconds, and a security solution takes 15 seconds, then this can be considered acceptable if the security solution brings added benefits and thus outweighs the drawbacks. That is why certain metrics will have so-called approximate results, i.e. the results should not be significantly different from each other. This also means that

these approximate results do not need the same kind of precision as exact results, as the added precision does not influence the outcome. As an example, it does not matter if the precision is 15 seconds or 14.89465767 if we are only looking at the precision of seconds.

3. Exact results: some metrics need more precise results to draw meaningful conclusions, in contrast to approximate results. For example, if we want to know how long it takes a router to validate a certificate, we want a precise answer as a small difference in results has an impact on the number of certificates that can be validated each second. An approximate result is simply not good enough here as we need the precision in these cases. Metrics that require exact results will be indicated and explained properly and require better precision than the approximate results that we talked about before.

Properly defining the kind of results that we gather in the simulations is important. Due to the difference between approximate and exact results, the time to run simulations decreases and allows for analysing and gathering other data in the given time frame of the project.

4.4. Scope

It is important to discuss and define the scope of this thesis before proceeding with the experiments and development of a solution. In the background we have discussed attacks against BGP, which also includes attacks other than community attacks. However, for developing the solution, we will only consider protecting the community attribute and thus only protect against community attacks. More specifically, we will focus on the 'normal' community attribute in this thesis, and not consider the large community attribute or the extended community attribute. However, extending the developed solution later in this thesis to also include these two different community attributes is trivial, but not considered for both readability of this thesis and quicker development and testing. The same goes for IPv6, we will only consider IPv4 in this thesis, but the translation to IPv6 should be trivial. Other BGP attributes are not relevant for this thesis and thus also not protected by the developed solution. Furthermore, vulnerabilities or weaknesses related to TCP, or any of the lower network layer protocols used by BGP, are out of scope. We will only protect a part of the control plane in BGP (specifically, the authorized usage of the community attribute), and not other attributes or the data plane.

Now that we have defined what we are protecting in BGP, we can further define the scope for the security solution that we are going to develop. This thesis is going to focus on mitigating community attacks via cryptographic authorization. We will do this by making use of RPKI objects, which will be further motivated and explained in Chapter 5.

In the background we also discussed that one of the issues of communities is the lack of semantics. It is important to note that semantics are not validated with this security proposal, but that this is an interesting improvement to the developed solution. This will also be further discussed in the discussion chapter.

At last, it is important to define what is achievable in this thesis, and what is not. In the time frame of the project, we will develop a theoretical security solution first. Afterwards, this solution is developed and tested in simulation software. The choice to do it in simulations is partly due to it being easier in the time frame than developing and testing on real hardware. On the other hand, due to the corona pandemic, the university was not able to provide the guidance and equipment for hardware testing, meaning that simulations are the only feasible option. Regarding the experiments to measure the metrics of the developed solution, this will be a split between theoretical analysis and actual measurements. Some properties, like scalability on the internet, are not possible to measure in simulations with only a laptop. However, we can analyse in theory how it would scale and use measurements on lower scales to see if we can estimate the scalability to internet size. We will further define and talk about the limitations of the approach and the analyses in the discussion chapter.

5 Design

In this chapter, we will discuss the solution design from both a theoretical and practical standpoint. We start by creating the theoretical solution and explain why it was chosen this way. Afterwards, we can theoretically analyze how it will perform and what it can protect against, but also what the potential drawbacks are. After that, we explain the practical implementation of the security solution together with drawbacks and improvements of the solution, the tools, simulation, and the implementation.

5.1. Theoretical

We start by discussing and explaining the theoretical part of the proposed security solution design. The solution is called RCA, which stands for Route Community Authorization, named after the related ROA in RPKI. We start by discussing the requirements for the solution design and the creative process to arrive at this solution. Afterwards, we discuss the newly developed RPKI object in detail, where it is explained what the capabilities are and why it works. Then, we discuss the deployment of RCA and its benefits. We end this section by discussing an example and looking at the drawbacks and improvements.

5.1.1. Requirements

As discussed in the background, there are a lot of requirements that a security solution for BGP should have to be considered viable for deployment. In the background, we discussed several metrics to assess a security solution. As a quick recap, a solution should be secure, performant, deployable, and privacy-friendly. But, as is the case for every single security solution, you cannot have all of them, there needs to be a compromise somewhere. Since we have seen that community values are widely used by ASes, the security solution should not impact the usability too much. Otherwise, ASes will not consider it if their usability is affected. Another important requirement is that the BGP protocol itself should not be altered.

The first idea that came to mind is signing the community value or attaching a Message Authentication Code (MAC) to each announcement. However, this would require cryptographic operations on every outgoing BGP UPDATE, which is expensive. We also know from the issues of BGPsec that cryptographic operations on every UPDATE message are not going to work with the current hardware and incentives of ASes. We need to consider a solution where cryptography can be offloaded, like RPKI and ASPA.

The second idea came to mind by doing just that. ASPA is interesting since it is a new RPKI object that promises AS-PATH verification. From the background, we know that RPKI objects are uploaded to a database, just like ROAs, so there is no cryptographic operation for each UPDATE message. Instead, the object defines customer-provider pairs that can be validated. ASPA has been claimed to be incrementally deployable, scalable, and similar in the computational requirements to RPKI. However, ASPA is still in the draft stage, and the integrity of the AS-PATH attribute itself is not protected. Check out Chapter 2 for more background and potential issues of ASPA.

Using this idea from ASPA, and looking at how ROAs work, we started working on the second idea. By introducing a new RPKI object, we can define which community values are allowed by signing them and uploading them as an RPKI object. When values are seen that are not included in the object, the UPDATE is dropped and a potential community attack is prevented. With this approach, there are still issues: since it is based on RPKI objects, it suffers from the same problems as ROAs and ASPA. There is no integrity and the objects are uploaded in global/regional databases, to name two specific examples. Privacy is impacted as well since everyone can look at which AS uploaded which allowed community values, which might expose sensitive information. Furthermore, ASes that are participating do not protect themselves but protect others that deploy RCAs. The more ASes participate, the more protected everyone becomes, but there are no security benefits if you are an early adopter.

The third idea is an iteration of the second idea, but slightly improved. 'Perfect' security is not always achievable, or even desired due to complexity or computational impact. A general good approach to securing things is to reduce the attack surface. With this, we mean that you define what is strictly allowed, and everything else is considered a violation. Just look at the examples that already exist in the wild: firewalls drop traffic which is not explicitly allowed, applications are hardened by explicitly listing what they can do and the Operating System stops everything outside of those capabilities (AppArmor is a good example on Linux[1]), ransomware protection on Windows lists which processes are allowed to edit files in protected folders, others are blocked[2], etc. What if we use this practice of listing what is allowed and consequently reducing the attack surface, and thus improve the security of community usage?

The third iteration uses this idea and adds more information to the RPKI object regarding specific conditions for community usage, which we will see in the next section in detail. By defining which ASes are allowed on the AS-PATH, the maximum AS-PATH length, regex for community values, max prefix length, etc. we can specify specific conditions where we can reduce the attack surface and thus achieve better security than the second idea.

The final idea is inspired by Peerlock and one of the critiques against ROAs. ROAs are stored in global databases, and global databases are accessible to any participating party. This means that sensitive information of an AS is required to be uploaded to a global database if the AS wants to deploy ROAs. A more privacy-friendly option is to exchange information between peers (which Peerlock does). This does not reveal sensitive information globally, which improves privacy. So instead of using global databases only, ASes can exchange RPKI objects with peers. The information that they wish to deploy globally can still be done, but by exchanging via specific peers certain information does not need to be exchanged with every single AS. When validating an incoming announcement, peer databases are prioritized, before looking in the global database. This approach has the benefits of both peer databases and a global approach.

5.1.2. RPKI Object

Now that we have defined the iterations of arriving at a generic idea for protecting community values, we can discuss the technical part and make it concrete. Below, we will list the complete definition of an RCA object and explain each of its properties. RPKI needs to be used by an AS before RCAs can be used.

Route Community Authorization

- version <int> (required): just like a ROA, this indicates the version of the object. It might change in the future, and this makes future upgrades possible.
- timestamp <int> (required): an epoch timestamp indicates when the RCA was created
- asn <int> (required): the ASN of the AS, matches with a ROA.

[1] `https://apparmor.net/`

[2] `https://docs.microsoft.com/en-us/microsoft-365/security/defender-endpoint/controlled-folders?view=o365-worldwide`

- validity_date_start <int> (required): epoch timestamp from which this RCA is considered valid.
- validity_date_end <int> (required): epoch timestamp from which this RCA is considered invalid.
- prefixes <array<ip» (required): array of prefixes, which are covered by a ROA, which this RCA covers. It can be a subset of the ROA prefixes.
- max_prefix_length <int> (optional, default 32): the maximum length of prefixes that can be used in route announcements to consider this RCA valid.
- ASes <array<asn» (optional, default allow all): array of ASNs that indicate which ASes are allowed to be included in the AS-PATH.
- as_path_length <int> (optional, default infinite): maximum number of ASes allowed in the AS-PATH attribute.
- communities <array<string» (required): array of strings that indicate the community values allowed for this RCA. Regex is allowed to add more flexibility.
- allow <bool> (required): indicates if community values listed before are allowed or should be denied. This makes it easier to create a deny policy whenever required by an AS.

5.1.3. Algorithms

There are two main algorithms to make RCA verification and validation work. The verification algorithm takes care of verifying the certificate of the RCA and makes sure that the RCA is considered valid according to the technical specification of the object. This offloads expensive operations, like cryptographic verification. When RCAs are downloaded periodically from databases, they can be verified once and uploaded to the routers, which then only have to run validation on announcements. This idea is similar to how ROAs are used in practice. The validation algorithm validates an incoming route announcement and matches it against one of the RCAs (if it exists). Depending on the match, the announcement is either allowed or dropped. We will list and discuss both algorithms below in Python.

```
# The verification algorithm for verifying RCAs. An RCA is embedded in a
 ↪ certificate. In addition, the matching ROA prefixes are fed to the
 ↪ algorithm, assuming that the ROAs have already been validated and
 ↪ verified. This algorithm is run when a new RCA is received and the
 ↪ result is 'cached'. Using this approach, the heavy verification that
 ↪ contains cryptographic verification is offloaded and improves
 ↪ performance on the router side.
def rca_verification(certificate, roa_prefixes, encr_alg):
    # verify the certificates based on the encryption algorithm that is
     ↪ used
    if encr_alg == "rsa":
        certificate.public_key().verify(
            certificate.signature,
            certificate.tbs_certificate_bytes,
            padding.PKCS1v15(),
            certificate.signature_hash_algorithm
        )
    elif encr_alg == "ed25519":
        certificate.public_key().verify(certificate.signature,
         ↪ certificate.tbs_certificate_bytes)
    elif encr_alg == "ecdsa":
        certificate.public_key().verify(certificate.signature,
         ↪ certificate.tbs_certificate_bytes, ec.ECDSA(hashes.SHA256()))

    # retrieve the embedded RCA and parse it into an object that can be
     ↪ used
```

```
    rca = parse_rca(certificate.extensions.get_extension_for_oid(
     ↪  x509.ObjectIdentifier("1.2.3.4")).value.value.decode())

    # if parsing fails, return False
    if rca is None:
        return False, "RCA parsing error"

    # verify that the RCA is still valid
    if datetime.datetime.now() > rca.validity_date_end or
     ↪  datetime.datetime.now() < rca.validity_date_start:
        return False, "RCA outdated"

    # loop through the prefixes of the rca and verify that they are
     ↪  covered by the ROA prefixes
    for prefix in rca.prefixes:
        covered = False

        for roa_prefix in roa_prefixes:
            roa_network = ipaddress.IPv4Network(roa_prefix["prefix"] + "/"
             ↪  + roa_prefix["prefix_len"])

            if ipaddress.IPv4Network(prefix).subnet_of(roa_network):
                covered = True
                break

        # if the prefixes are not covered by ROA, the RCA is not valid
        if not covered:
            return False, "RCA prefixes not covered by ROA prefixes"

    # RCA verification succeeded
    return True, "Valid RCA Certificate"
```

```
# The validation algorithm that is run on each incoming announcement. RCA
 ↪  verification is offloaded, so it is assumed that any RCAs fed to this
 ↪  algorithm are verified. The algorithm needs the announcement and an
 ↪  array of RCAs that match the origin AS. This array first contains the
 ↪  rcas received from a peer (if any) and then the global rcas (if any),
 ↪  meaning that peer RCAs are considered a higher priority than global
 ↪  RCAs.
def rca_announcement_validation(announcement, rcas):
    # gather the array of communities from the announcement
    community_list = announcement["community_list"].split()

    # If the announcement does not contain communities, it is
     ↪  automatically valid.
    if len(community_list) == 0:
        return True, "Announcement does not contain communities"

    # gather the array of ASes on the path from the announcement
    as_path_list = announcement["as_path"].split()

    # get the network that is announced
    announced_network = ipaddress.IPv4Network(announcement["prefix"] + "/"
     ↪  + str(announcement["prefix_len"]))
```

```
    # loop through each RCA in the array and validate if it covers the
     ↪ announcement. When an RCA matches, return True. Otherwise, go to
     ↪ the next one until we looped through the array. In that case, the
     ↪ announcement is not covered by the RCAs and is thus invalid.
    for rca in rcas:
        covered = False

        # verify if the prefixes in the announcement match the prefixes in
         ↪ the RCA, otherwise continue
        for prefix in rca.prefixes:
            if announced_network.subnet_of(ipaddress.IPv4Network(prefix)):
                covered = True
                break

        if not covered:
            continue

        # validate if the prefix length does not exceed the max prefix
         ↪ length of the RCA
        if rca.max_prefix_length is not None and
         ↪ announced_network.prefixlen > rca.max_prefix_length:
            continue

        # validate if the ASes of the announcement are a subset of the
         ↪ allowed RCA ASes. Otherwise, the announcement contains ASes
         ↪ not allowed and is thus not covered by the RCA.
        if rca.ases is not None and not
         ↪ set(as_path_list).issubset(set(rca.ases)):
            continue

        # validate if the AS path length is not exceeded
        if rca.as_path_length is not None and len(as_path_list) >
         ↪ rca.as_path_length:
            continue

        # when parsing RCAs, a regex is built where the communities are
         ↪ ORed together like this:
        # - RCA contains communities 666, 103, and a regex expression X
        # - the validation regex becomes "666|103|X"
        # using this approach, we can match each announcement community
         ↪ against the regex validation string, even if one of the RCA
         ↪ communities is defined by a regex.

        match = True

        # loop through each community and match it against the regex
         ↪ string.
        for community in community_list:
            if rca.regex.search(community) is None:
                match = False
                break

        # if a community is not covered by the regex, continue to the next
         ↪ RCA
        if not match:
            continue
```

```
        # if the communities match the RCA, but the communities are
          ↪ disallowed by the RCA, immediately return False for the
          ↪ validation algorithm
        elif not rca.allow:
            return False, "Announcement disallowed by RCA: " +
              ↪ rca.identifier

        # announcement matches the RCA and is allowed, return True
        return True, "Announcement covered by RCA: " + rca.identifier

    # none of the RCAs matched the announcement, return False
    return False, "Announcement not covered by any RCA"
```

5.1.4. Deployment

Now that we have a theoretical basis for the security solution, it is time to plan a deployment strategy. Since our goal is to not alter the BGP protocol, there are two options:

1. Routers need to add support, just like with RPKI and other protocols like BGP Monitoring Protocol (BMP), which we will see later. These are protocols that do not alter BGP itself but add additional functionality. However, altering a router is not a reachable goal for this thesis. Another option is altering a BGP software router. A few good open-source projects are FRRouting, BIRD, and GoBGP. However, altering these open source projects takes a lot of time as well, which is not achievable in the time frame or part of the goal of this thesis. An advantage of a router solution is the ability to drop invalid routes directly with RCAs, something that is not achievable with the next option.

2. Monitoring the BGP messages, like ARTEMIS does with their BGP hijacking protection[3], and responding to detected attacks to mitigate them with a specifically crafted BGP UPDATE message. To achieve this, we need to be able to monitor incoming BGP messages in the Adj-RIB-In. Luckily, there exists an excellent protocol that has been developed in 2016 which does exactly that: "BGP Monitoring Protocol (BMP) is intended to provide a convenient interface for obtaining route views. Before the introduction of BMP, screen scraping was the most commonly used approach to obtaining such views. The design goals are to keep BMP simple, useful, easily implemented, and minimally service affecting." [38]

   BMP is standardised in RFC 7854 and is supported by most major vendors like Juniper, Cisco, GoBGP, and FRRouting (both hardware routers and software daemons) for multiple years. We can use BMP on routers where we want to use RCA to stream the incoming BGP updates to a server. This server uses the validation algorithm for all incoming announcements and handles the retrieval and verification of RCAs. If a violation is found, it is the task of this server to mitigate the potential attack. Note that we use BMP in monitoring mode and not mirroring mode. "In BMP's normal operating mode, after the BMP session is up, Route Monitoring messages are used to provide a snapshot of the Adj-RIB-In of each monitored peer" [38], while mirroring mode duplicates all BGP messages received. Mirroring is not an option since it can lead to "router resource exhaustion, the potential to interfere with the transmission or reception of BGP UPDATE messages, and the slowing of routing convergence" [38].

Now that we have a deployment approach which is minimal service affecting and widely usable, we are just left with two tasks: RCA validation needs to be able to keep up with the stream of incoming BGP updates on the internet (i.e. it needs to be performant), and we need to be able to remove violating routes from the router's routing table.

The first task is something that we can analyze in the results, so it is not relevant for now. Dropping invalid routes is challenging since you cannot withdraw a route that you did not announce in BGP. Something that we can do, however, is overwriting the invalid route with a 'cleaned' route. In the BGP

[3] https://bgpartemis.org/

best-path selection algorithm, the first step is to always prefer the route with the highest local preference value[4]. A strategy that we can use to overwrite routes is easy and quick:

- Remove invalid communities that are not covered by RCAs from the announcement.
- Set the local pref value to be 1 higher than the announcement itself.
- Announce the route to the router.

Using this strategy successfully overwrites the route, which can be seen in Chapter 6. The routing table uses the mitigation route instead of the violating route.

5.1.5. Example

Let's demonstrate the capabilities of RCA using an example. For this example, we will be using an RCA object defined in Table 5.1.

RCA attribute	Value
Version	1
Timestamp	1651841017
ASN	3
Validity_date_start	1651841017
Validity_date_end	1683369885 (∼1 year later)
Prefixes	[3.3.3.0/24]
Max_prefix_length	32
ASes	[2, 3]
AS_path_length	2
Communities	[\d+:666]
Allow	True

Table 5.1: Example RCA object for AS 3

In plain text, this means that AS 3 has issued a certificate that is valid for one year, which authorizes other ASes to use the community value \d+:666 (regex that allows any number, followed by :666, note that this is the blackhole community) with the following conditions: this community value can only be used for the IP addresses within 3.3.3.0/24, can be as specific as /32, only ASes 2 and 3 are allowed in the AS-PATH, and the AS-PATH cannot contain more than 2 ASes.

A different example might be a disallow RCA. If using the above example, but we change allow to False, we disallow announcements that match the attributes from using the community value.

5.1.6. Attacks

In this section, we will show the relevance of each attribute in an RCA that might prevent an attack. We have described that we are reducing the attack surface by specifically mentioning what is allowed and disallowed. Now, to get a better understanding of why this is important, we will demonstrate an example for each attribute that can stop an attack below:

- validity dates: An attacker AS is not able to use a community since its use has expired.

[4]Except in Cisco routers, where it is the second step

- prefixes: An attacker AS cannot use a community for a prefix that is not specified in the RCA.
- max prefix length: An attacker cannot use a more specific prefix than allowed by the RCA.
- ASes: If an RCA only allows AS 3 in the path, then the community value becomes invalid when any other AS is in the path besides AS 3. This limits the reach of the community value and by which ASes the community can be used.
- AS-PATH length: An AS cannot use the community value more than a certain number of ASes away from the origin (useful when community values are only used by direct peers of an AS for example).
- communities: An attacker AS cannot tag the announcement with a community that is not allowed.
- allow: An attacker AS cannot use community values if the values are disallowed by the RCA.

Besides stopping community attacks, RCAs can also solve one of the issues that RTBH has, which was discussed in Section 2. It was shown that prefixes are blackholed longer than they should, or even forgotten about. By using RCAs, you could upload an RCA that is only valid for one day for example, and after it expires the blackhole value can no longer be used. This also means that providers do not have to always allow blackhole communities on their prefixes, but simply upload a short-lived RCA when needed. This can prevent abuse of blackholing.

One might think that this causes a big delay before an RCA has effect, since RCAs are fetched every X minutes. However, note that there are peer databases that we can make use of in this case. If we need to authorize a blackhole to drop traffic to a specific prefix, we can upload an RCA to the peer databases of these ASes, and it immediately takes effect (as peer RCAs are directly uploaded and received periodically). The delay of a global database fetch is circumvented with this approach, and it is another advantage of using peer databases besides the privacy benefits.

5.1.7. Conditions

In the background, we discussed the necessary and sufficient conditions for an AS to be able to successfully perform a community attack. We can also derive the necessary and sufficient conditions for defenders to successfully mitigate a community attack, which we will list below:

1. Necessary condition: Community attacks can be mitigated if the origin has configured their RCAs correctly to block the community value that an attacker uses for the attack.
2. Sufficient condition: For community attacks to be mitigated, it is sufficient if the target AS, or a single AS on the path between the attacker and the target, deploys RCA.

Do note that due to certain RCA configurations, attacks can only be stopped at the target AS and not by an AS on the path between the attacker and the target. Correct RCA configuration is therefore important. Furthermore, malicious ASes could modify the AS-PATH attribute for example to circumvent an RCA if that RCA relies on the ASes and AS-PATH length attribute. We therefore recommend deploying both ASPA (when it becomes available) and RCA if possible.

5.1.8. Drawbacks and improvements

The first drawback of the current approach is that there is still no integrity. However, adding integrity protection requires cryptographic operations per announcement, which is not realizable yet with the hardware deployed at ASes. Maybe in the future this is possible, but we are trying to make a viable solution that is deployable with current hardware.

The second drawback is that this system is susceptible to human error. If an AS specifies its community values too specific, the usability is impacted. However, if an AS makes it too broad, an attacker can abuse the allowed values that it still has access to. This is very similar to, for example, configuring firewalls. It can be solved by teaching the operators about the proper use of RCAs.

There is still room for improvement however. The current attributes are just examples of what we

came up with that can protect community values by limiting the attack surface. Maybe it is necessary to add more attributes, like minimum prefix length, to make it more usable for ASes. The idea of this research is not to make this specific version of the RCA and not allow any changes, but to show the potential and the ability to stop a new kind of attack against BGP. If further testing and input from the community show the need for different attributes, this can of course be added or modified appropriately (or added in a new version update).

Last, but not least, currently we are adopting an approach of dropping invalid routes. However, this could introduce a new attack with RCAs. If a malicious AS purposely makes use of communities that are not allowed, it can cause a target AS that is either one or more hops away (depending on if the ASes between the attacker and the target remove the communities or not) to not accept otherwise valid routes. This can cause connectivity issues or higher economic costs by using more expensive routes for example. An improvement to RCAs could be to not drop routes with invalid communities, but clean the announcements instead. An invalid community is removed from the announcement, and the rest of the announcement is accepted by the AS. This mitigates the attack that we described. Another possible attack is that since you can specify regexes for RCAs, it could be the case that a misconfiguration or a malicious AS causes an RCA to contain a regex that takes long to run. To combat a potential DoS where a regex takes too long, you can specify a maximum amount of time that each announcement validation can take. Any malicious attempts will be obvious as all ASes can see where the RCA came from, so this could also discourage anyone from constructing malicious regexes.

5.2. Implementation

After discussing the theoretical part of an RCA object, we can now take a look at the implementation. We start this section by giving a detailed overview of the implementation of RCA. After that, we discuss the architecture, where we give an overview of what RCA deployment looks like. We end the section by discussing an example and looking at the drawbacks of the practical implementation.

5.2.1. Overview

The first step for a practical implementation, where we can test RCAs with BGP routers, is an environment for network simulation. We decided to use GNS3 for this as it was recommended by one of the professors and it is free to use.

Next, we need to run the actual routers. Since the amount of RAM and CPU cores is limited on the laptop where the experiments are running, we need a small image that GNS3 can run and still has all of the functionality that we require. Alpine Linux images are great for this, as they are lightweight. The images are using the qcow2 format so that we can use QEMU for virtualization which allows the GNS3 project to be portable. We decided to go with GoBGP as the BGP router since it is open source. The Linux images are running GoBGP with Zebra to enable kernel routing.

Now that we have network simulation, virtualization, and BGP routing, we need to set up the environment for RCAs. The first step is to use RPKI. We can use Krill for a local RPKI certificate authority and Routinator to fetch and distribute verified data via the RPKI-to-Router (RTR) protocol to the routers. These two programs are running on a server in the simulated network to which the routers can connect to.

With RPKI running, we need to use BMP. BMP requires a BMP server to receive the data from the routers. Luckily, there is an open-source project that implements this server called OpenBMP. With OpenBMP, there is great support for languages to connect to the server and receive the stream of BGP messages. For this project, we decided to use Python as it has great support with the technologies that we use and allows for quick and easy scripting. Using Python, we can connect to the BMP server and process each announcement. In Python, we have also implemented the RCAs together with the necessary algorithms to validate and verify incoming RCAs and announcements.

Next up for RCA support is a global database and a peer database per AS to store RCAs. These two databases are also implemented in Python. ASes themselves can connect with other ASes to dis-

tribute and receive RCAs whenever desired, or store them in a global database that is fetched every X minutes. We also need a BGP speaker to be able to announce the mitigations to the routers if a violation is detected. We decided to use ExaBGP for this, as it also uses Python and is a quick and easy way to announce custom BGP messages in a user-friendly plain text format.

Last, but not least, we need to encode the RCA in a certificate, just like ROAs. To make this work, we should make a custom ASN.1 encoding to be able to add it as an extension to an x509 certificate. Since that takes a lot of time, we simply encode an RCA to a string. This string can then be embedded in an x509 UnrecognizedExtension in Python. In our experiments, we can still use this unrecognized extension in supported libraries in Python to embed an RCA in a certificate without going through a time-consuming process to create custom ASN.1 encoding to test certificate verification with RCAs. By simply encoding an RCA to a string, embedding it in a certificate, and decoding it from the string upon verification of the certificate, we have a fully working certificate solution to test.

5.2.2. Architecture

Now that we have implemented all necessary technologies and systems to have RCA support for routers, we create and explain an architecture diagram to have a good idea of what the deployment of RCA currently looks like. Note that if routers decide to implement RCAs in the future, the RCA servers in the architecture are no longer needed. Please look at Figure 5.1 for an overview of the RCA deployment architecture in an example with three ASes, where AS 2 is malicious. Note that the BMP server, ExaBGP speaker and RCA scripts (together with a peer database) are all included in the RCA Server.

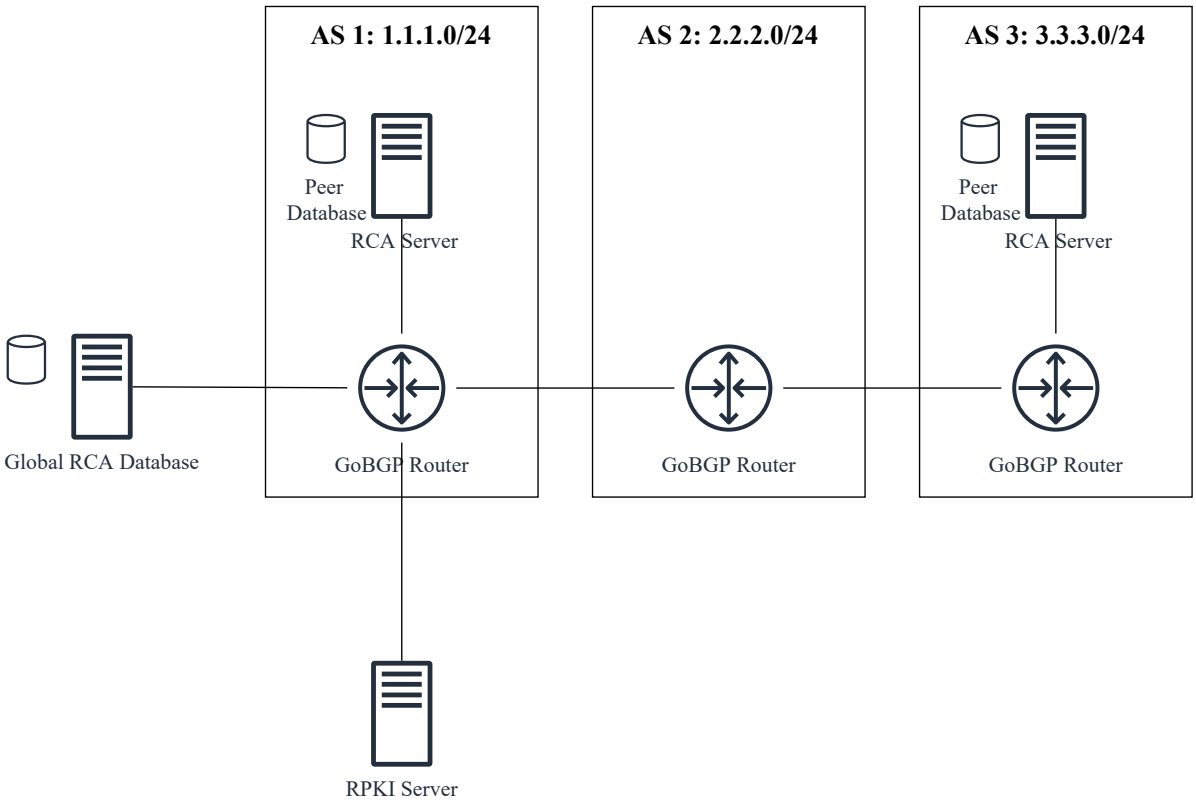


Figure 5.1: Architecture diagram of a simple RCA deployment. Note that the BMP server, ExaBGP speaker and RCA scripts (together with a peer database) are all included in the RCA Server.

5.2.3. Example

Using the architecture diagram from above, we will discuss an example to explain the flow of the inner workings with RCA. This gives a good idea of how it works internally and how everything flows between the different systems. We explain and show the process of registering RCAs, up to mitigating an at-

tack in the overview below. The steps reference the systems that are shown in the architecture diagram.

In the example that is described below, AS 2 is considered malicious. AS 3 is trying to protect its prefixes by deploying RCAs, just like AS 1. This means that both AS 1 and AS 3 can protect each other. All ASes use RPKI, but in this case it is only necessary for AS 1 and 3. Furthermore, AS 1 implements a blackhole policy, which AS 2 will try to abuse by blackholing a prefix of AS 3.

1. AS 3 crafts the RCA to protect its prefixes from blackhole attacks.
2. AS 3 signs an x509 certificate that contains the embedded RCA.
3. AS 3 has the option to either upload the certificate to AS 1 directly or in a global database, depending on its privacy needs.
4. AS 1, or the global database, runs the verification algorithm before accepting the RCA and storing it in the database. If the RCA is stored globally, AS 1 will retrieve it during a periodic fetch and rerun the verification algorithm before adding it to its internal copy of the global database.
5. AS 2 decides it is time to try a blackhole attack. After AS 3 announces the IP range 3.3.3.0/24, it tags the announcement with 666 (Note, normally blackhole is only allowed for /32 ranges, but this is just for demonstration purposes. Blackhole attacks are easier to understand than other community attacks).
6. AS 1 receives the tagged announcement from AS 2. RPKI validation succeeds and the route is added to the RIB. Since the blackhole policy matches with the community value, AS 1 drops traffic to 3.3.3.0/24.
7. BMP monitors the incoming BGP UPDATE and sends it to the OpenBMP server.
8. The OpenBMP server streams the BGP UPDATE to the listeners, in this case, the RCA validation script.
9. The validation script checks if it has any RCAs for AS 3 and the prefix 3.3.3.0/24, which it has, and runs the algorithm.
10. The validation fails because AS 2 is not allowed on the AS-PATH (or a different approach to blocking the blackhole attack, like completely disallowing 666).
11. The script notifies the ExaBGP script to mitigate the attack.
12. ExaBGP announces the same announcement, but with the community removed and with a higher local preference to overwrite the route.
13. AS 1 stores the route in its RIB and the route of AS 2 is overwritten due to the higher local preference, mitigating the blackhole attack.
14. Traffic to 3.3.3.0/24 is restored.

5.2.4. Drawbacks

The deployment strategy from above has many upsides. BMP is performant and does not introduce noticeable performance impact, the router does not have to handle the additional load of validating communities due to BMP, BGP itself is not altered, and routers that support BMP and RPKI can use this security measure already by simply deploying an additional server, to name a few examples. However, this approach also has drawbacks of course.

The first drawback is that RCA is not running on the router itself. There is a small time frame where the attack is successful and where traffic in the above example is dropped. We will analyze the consequences of this in Chapter 6. This is because in a BMP deployment, routes are accepted initially and the BMP server processes the received routes shortly afterwards. When a violation is seen, then it is mitigated. Routers will still receive all routes and all communities when using BMP and receive mitigations shortly after the initial receival of routes.

The second drawback is that RPKI needs to be used. An AS that does not want to run RPKI, but wants to use RCA, is not able to use it. We will further discuss any drawbacks that arise during the simulation and from the experiment results in Chapter 6, where the results are discussed and analyzed.

6

Evaluation

In this chapter, we discuss how the experiments in the simulations are performed and the setup for the experiments to gather the required data for this thesis. In some cases, it is not possible to measure the required data in the simulation setup, which is why have to make some theoretical analyses. We first provide a disclaimer about the collected results in this chapter. Then, we start by discussing the theoretical results. In addition, using the experiments later on, we can confirm some of the claims that we have made so far regarding the capabilities of the developed solution. Furthermore, we discuss the approximate results, as defined in the methodology chapter, and the exact results. At the end, we create a threat model to analyze what damage an attacker can do during the short time until the mitigation is sent when using BMP.

6.1. Disclaimer

The results from the experiments gathered here were gathered on an HP ZBook G4 Studio with an Intel Core i7-7700HQ and 24 GB RAM. The simulations were run in GNS3 2.2.31 with the GNS3 VM running on VMWare Workstation 16.2.3 Player. The BGP routers are using GoBGP v3.0.0-rc3 on Alpine Linux 3.15. The RCA implementation was made in Python 3.8. Running the experiments on other hardware or software can result in different results.

Furthermore, the algorithms and implementations that are shown have not been optimized and are written in Python. After optimization and implementing it in different languages, results can and will probably be better.

Last, but not least, due to the restrictions of this thesis, all experiments were done via BMP monitoring deployment as explained in Chapter 5. There are advantages and disadvantages of running on a router directly, or via the BMP approach here, we will also see in this chapter.

6.2. Theoretical results

As we have seen previously, we need to judge our RCA solution based on the following properties:

- Security properties
- Performance properties
- Privacy properties
- Deployability properties

As some of the properties can only be analyzed theoretically, we will discuss each of them below. Furthermore, for each of the properties that cannot be measured, an appropriate explanation will be given.

6.2.1. Security properties

When assessing the security properties of a solution, we need to know how much extra security it guarantees, which vulnerabilities remain for BGP, and if new attack vectors are introduced. We will discuss each of these three below, but these properties cannot be expressed or measured precisely.

The extra security is a difficult property to assess. As we have seen in the solution design, RCA makes sure that each announcement uses communities that have been authorized to be used by the origin AS. The added security depends on the configuration of the origin AS. When properly configured, RCA reduces the attack surface to only those values that have been defined by the origin AS, just like a firewall configuration only allows traffic that is configured by the administrator. However, to make the security work, multiple ASes need to deploy RCAs to protect each other. And this is also where incentive becomes difficult. If only one AS deploys RCAs, then there is still no security. Just like with RPKI, you can only enforce the security when multiple people use it. This is also the biggest issue of RCA, because ASes have misaligned incentives here, you want to protect yourself not others. We will discuss this issue further in Chapter 7. If two ASes deploy RCAs, then they will only protect each other. Depending on if they are neighbours or not, this protection is either useful or not. Once people start to pick up RCAs more, the security for everyone increases. RCAs thus offer incremental security once more people start using them, but there is a slow start just like with RPKI and ASPA. Using different experiments that we see later in this chapter, we know that if RCAs are properly configured, and ASes deploy the solution, then community attacks are correctly mitigated. We have also seen in Chapter 5 that the sufficient condition for mitigating community attacks is that only a single AS between the attacker and the target, or the target itself, needs to deploy RCA for successful mitigation, assuming that the RCAs have been configured correctly to detect attacks.

Due to the approach we took for RCAs, we only guarantee authorized use of communities. An AS lists the conditions for the communities that can be used, and everything else is not allowed. This means that misconfigured RCAs (for example, an RCA that allows almost everything) can still be exploited in attacks. Furthermore, RCAs only protect community usage in BGP, other attributes are not protected or affected. Any other issues in BGP apart from what RPKI solves and what RCAs can guarantee are not protected and remain an issue. As an example, the integrity of community values is not protected, meaning that ASes can alter the values without the receiving party knowing if something was changed (as long as the changed value is allowed by an RCA). The AS-PATH attribute can still be altered as well, which might enable attackers to circumvent an RCA that relies on the ASes or AS-PATH length attribute as explained in Chapter 5. We thus recommend to also deploy ASPA once it is available.

As far as we know, RCAs do not introduce new attack vectors, apart from the one listed in Chapter 5. Since the approach that we took is 'just' limiting what can be done with community values, we do not introduce new behaviour in BGP, it only states what can no longer be used. Therefore, there is not a lot of room to introduce new vulnerabilities. As seen in the attack of Chapter 5, one might argue that a malicious AS could modify announcements and include communities which will cause the route to be dropped at the target AS. This potential attack vector can prevent a target from accepting new routes. A solution to this is to apply a 'cleaning policy' instead of dropping routes. By removing the invalid communities upon violation, routes are not dropped and this attack is not possible anymore. In addition, there is one issue with RCAs which is explained by analyzing the threat model. When deploying RCAs and mitigating attacks via BMP, a malicious announcement is still accepted by the router for a brief moment. How long this moment is exactly, and what damage can be done during this moment, will be analyzed further in the threat model later in this chapter. Lastly, since you can specify regexes for RCAs, it could be the case that a misconfiguration or a malicious AS causes an RCA to contain a regex that takes long to run. To combat a potential DoS where a regex takes too long, you can specify a maximum amount of time that each announcement validation can take. Any malicious attempts will be obvious as all ASes can see where the RCA came from, so this could also discourage anyone from constructing malicious regexes.

6.2.2. Performance properties

To analyze the performance of BGP, we can use a combination of experiments and analyses. Especially when not deploying and testing these solutions globally on the internet, but only in simulations, certain

data cannot be gathered. We will discuss each of the performance properties below.

- Convergence delay: we can measure if using the security solution causes BGP to take longer to reach a stable state than using plain BGP. Exactly measuring this is possible but not that meaningful without real hardware routers and more than a few routers in software. You want to measure the convergence with internet-like circumstances. To compensate for this, we will measure properties like scalability with performance benchmarks instead, and analyse in theory if the solution will cause stability issues, as seen below in the next point. If we can theoretically predict that the stability is not influenced, and show with benchmarks that the overhead on the routers is reasonable when scaling and can keep up with the rate of announcements seen on the internet, we can predict that the convergence delay will not be impacted greatly. However, exact numbers need to be measured in the future to confirm these theories, but this is left for future work. This will be discussed further in Chapter 7.

- Stability: without deploying RCAs globally on the internet, this is a difficult property to measure. RCAs themselves should not cause instability from our testing, since we are only stating which community values can be used or not. If an AS uses values that we disallowed, then the route will be dropped at an enforcing AS. Thus, instability can happen if ASes try to use community values which are not allowed. This can be prevented with the cleaning policy that we discussed against this attack.

- Scalability: this property is also difficult to measure without deploying it on the internet. Besides the approach that we discussed for the convergence delay, we can additionally analyze how much longer validating each route takes. Using both of these data, we can predict if RCA usage will scale or not.

- Computational overhead: just like with scalability, we can measure the number of announcements that can be processed per second and determine what the overhead is, and if this is an issue or not for BGP. We can estimate how many routes can be processed per second, and see how many routes are usually processed per second on the internet by a single router.

- Bandwidth overhead: additional bandwidth is needed to exchange RCAs globally and between peers. The overhead depends on the size of the database that needs to be exchanged. In addition, if BMP is used, then the the Adj-RIB-In is sent to the BMP server, which increases bandwidth. However, there is no additional bandwidth required in the BGP protocol other than any mitigations that are sent, since all of the communication is done out of band. We can calculate how much space an RCA takes, and calculate an estimate based on that.

- Storage overhead: using the storage requirement for a single RCA, we can calculate an estimate for how much extra storage is required when using RCAs with certain parameters (number of communities, prefixes, etc.) and at a certain deployment stage (i.e. 10% ASes use RCA, 30%, 50%, etc.).

For each of the parameters in the validation and verification algorithms, which mainly determine the performance impact of RCAs, we can calculate the impact of increasing the number of communities and prefixes. This will also reveal how the developed algorithms scale (e.g. exponential, polynomial, or linear).

6.2.3. Privacy properties

Because RCAs allow exchanging certain RCAs only between specific peers, the protocol does not have to reveal additional sensitive information compared to plain BGP. If ASes use global RCAs however, then privacy can be impacted. If properly used, then this should not be a problem.

6.2.4. Deployability properties

We have seen in implementation of the solution design, that deploying RCAs itself is not necessarily difficult. Although extra hardware is needed, it is only one additional computer which can be a normal laptop or a simple server, it does not need a lot of computational power as we will see later in the experiment results. Furthermore, you only need to use RPKI if you did not already, install the RCA scripts,

configure the RCAs (both on a global and on a peer level if desired), and connect the server via BMP to the router. That is all, the most difficult step is configuring the RCAs or deploying RPKI if it was not already used before. Routers without BMP are not supported, but we have seen that the major router vendors do ship routers with BMP support since a few years. If in the future routers support RCA, then deploying it will be easier since BMP and manually installing scripts is no longer required.

As discussed in the security properties above, the security of the solution is incremental with the number of ASes deploying it. Just like with RPKI, this means that the security guarantees are low when adoption is low. We know that at least the target itself, or one of the ASes on the path between the attacker and the target, needs to deploy RCA to mitigate attacks when the origin AS has configured their RCAs correctly. Thus, as with RPKI, security benefits will increase when more ASes join. This can cause an issue with incentive though, as ASes want to protect themselves immediately. Adoption is the biggest issue for RCAs after all, which will be further discussed in Chapter 7. Communication is done out of band though, meaning that BGP itself does not need to be altered.

6.3. Approximate results

In this section, we will discuss the approximate metric results. As explained in Chapter 4, these are results where the outcome should just show that the result is approximately in the neighbourhood of other results, or has a significant impact. We will explain the different experiments and how they are conducted below.

6.3.1. Time To Mitigation

Since the experiments are running in the BMP deployment instead of router deployment, attacks that are detected are not immediately dropped. This means that there is a time window where the attack is successful until the mitigation is launched. Using a threat model, we will later show what an attacker can do within such a time frame. Time to mitigation is measured in the following way with the architecture shown in Figure 5.1:

1. The attack is launched on router 2 on the prefixes of router 3. Router 3 is disabled due to RAM size restrictions on the laptop, but router 2 can still announce its prefixes without influencing this experiment.
2. When the malicious announcement is seen on the wire (we use Wireshark[1] for this), the exact time is noted.
3. When the mitigation announcement is seen on the wire, the exact time is noted.
4. The difference between these two times is the time to mitigation and shows how long a malicious announcement is accepted in the monitoring deployment before it is overwritten by the mitigation.
5. The measurements are repeated for both the Global and Peer database to see if either of the database deployments show a significant difference.
6. The measurements are repeated both with a simple BGP table (only contains the routes from the ASes in the architecture diagram) and full BGP tables (contains all routes that a typical router will contain when deployed on the internet, about 950k at the moment of writing). Full BGP tables can help indicate if there is an obvious bottleneck in mitigation time depending on the BGP table size, as this is the size of a router's table when deployed live on the internet. We injected a BGP table from the global route collector RRC00 from RIPE RIS in Amsterdam[2]. We wait until BGP has converged before starting the experiment.

The reason this experiment uses approximate results is that we are running it in simulation software. The results will be different in the real world and across different hardware (or software). Ideally, you would want to run RCA on a router directly anyways, which means there will be no time window where

[1] https://www.wireshark.org/

[2] The data can be downloaded here: `https://data.ris.ripe.net/rrc00/`. Note that at the time of reading this thesis it might contain more or fewer routes than at the time of testing.

attacks are successful. The second reason for approximate results is that we try to see if there is a significant difference between global/peer databases or simple/full BGP tables. A significant difference can be seen in this experiment setup, but we are not looking at differences of milliseconds here. The median of the results can be seen in Table 6.1, and the boxplot of the data can be seen in Figure 6.1. This experiment was run 10 times since we are only interested in an approximation of the mitigation time, not a precise number.

Deployment type	Median
Global RCA database with simple BGP tables	0.11175704002380371 seconds
Global RCA database with full BGP tables	0.11802792549133301 seconds
Peer RCA database with simple BGP tables	0.115181565284729 seconds
Peer RCA database with full BGP tables	0.12220454216003418 seconds

Table 6.1: Time to mitigation results

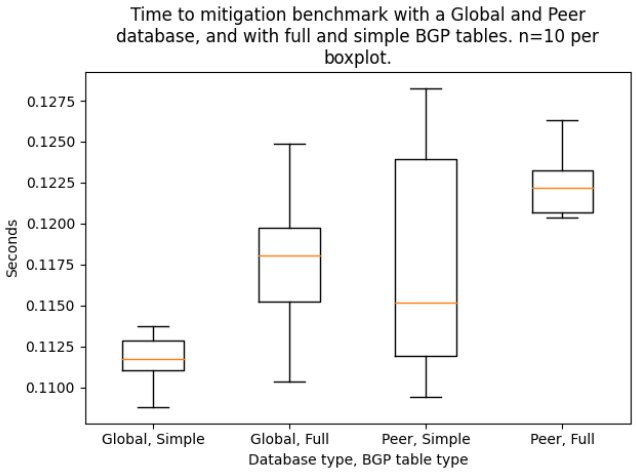


Figure 6.1: Boxplot of the time to mitigation results.

From the boxplot in the figure, we cannot conclude statements like a peer database or global database is faster. It does seem like full BGP tables make the mitigation time a bit slower, which would make sense. However, more experiments are needed to confirm conclusions like that, this is not the purpose of the mitigation time experiment since we do not see a significant differences. This experiment shows that the median mitigation time is relatively close to each other.

A mitigation time of about 0.12 seconds sounds good, but we need to see what it actually means. Therefore, it is important to look at the damage that can be done in this amount of time by an attacker to see the impact. This will be analysed at the end of this chapter in a threat model.

6.3.2. Average BGP Data

In this 'experiment', we gathered average BGP data from multiple sources. This data can be used later to draw meaningful conclusions about the performance of the algorithms, or estimate the average attributes that need to be present in an RCA and the announcements that we will use for the benchmarks later in this section. The collected statistics can be seen in Table 6.2.

6.3.3. RCA size

To estimate the bandwidth and storage overhead, we first need to estimate the size of an RCA and the certificate it is embedded in. An RCA is encoded by separating each attribute of the object with a '|'. We

Data	Average
AS-Path length	5 [39]
Announcements per day for a router	peaks as high as 1 million IPv4 announcements [40]
Communities per announcement	~90% of announcements contain ≤ 10 communities [8]
Daily convergence time IPv4	50-60 seconds [40]
Number of ASes in routing system	73444 [22]

Table 6.2: Average BGP data

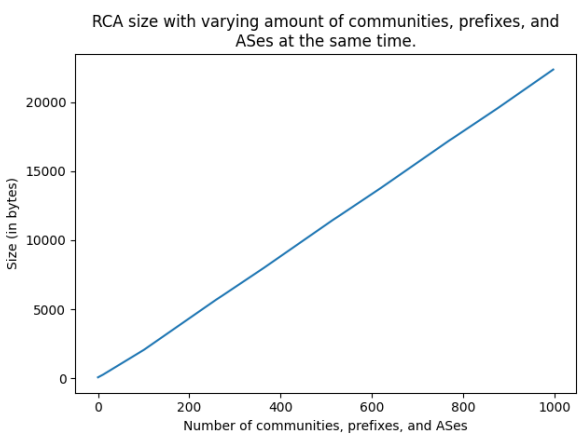
calculate the following RCA and certificate sizes in Table 6.3 under different amounts of communities, prefixes, and ASes.

Object	Size
RCA size 1 community, 1 prefix, and 1 AS	68 bytes
Certificate PEM size with RCA embedded 1 community, 1 prefix, and 1 AS	522 bytes
RCA size 10 communities, 10 prefixes, and 10 ASes	222 bytes
Certificate PEM size with RCA embedded 10 communities, 10 prefixes, and 10 ASes	733 bytes
RCA size 100 communities, 100 prefixes, and 100 ASes	2023 bytes
Certificate PEM size with RCA embedded 100 communities, 100 prefixes, and 100 ASes	3179 bytes

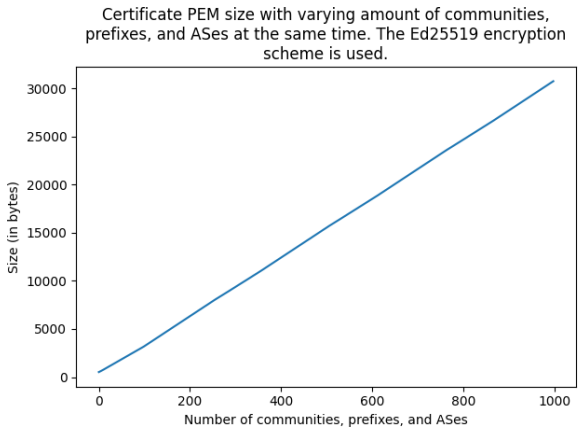
Table 6.3: RCA and certificate size

Of course, depending on how much information the RCA contains, the size will either increase or decrease. We encourage each AS to calculate the sizes based on their community usage and prefixes they want to cover to get a better idea. In Figure 6.2, we plotted the growth of the RCA size when varying the number of communities, number of ASes, and the number of prefixes at the same time. The figure shows that the size scales linearly. Size growth is of course very dependent on each variable. An extra AS in the allowed ASes, for example, is just an extra number. An extra community, however, depends on how the community is defined. ”1:666” does not grow the size much, but a complex regex would have a bigger impact. In this experiment we assumed simple communities. We cannot make concrete statements about RCA size under each circumstance, but we can show that the RCA size scales linearly and how that influences the size of certificates, which is also linear.

If we assume for the following example that each AS that uses RCAs has an average of 10 RCAs, each containing 10 communities, 10 prefixes, and 10 ASes, we can calculate the requirements per AS storage-wise. The number 10 only serves as an example here to show storage and bandwidth estimations. We consider that 30% of total ASes deploy RCAs in the following calculations, which can be seen in Table 6.4. At the time of writing, a total of 73444 ASes are in the routing system. Since we



(a) RCA size when varying number of communities, prefixes, and ASes.



(b) Certificate PEM size when varying number of communities, prefixes, and ASes.

Figure 6.2: Plots of how the RCA and certificate size grow when varying the number of communities, prefixes, and ASes at the same time.

have shown in Figure 6.2 that RCA size grows linearly, the storage and bandwidth requirements will do the same.

Storage	Size
Global Database Total (stores RCA certificates)	$10 * 733 * 0.3 * 73444 = 161.503356$ MB
Storage per AS (only encoded RCAs)	$10 * 222 * 0.3 * 73444 = 48.913704$ MB
Bandwidth downloading Global Database every X minutes (RCA certificates)	161.503356 MB every X minutes
Bandwidth uploading to Global Database (RCA certificates only when new)	733 bytes per certificate upload

Table 6.4: Storage estimations in a 30% AS deployment with 10 RCAs each containing 10 communities, 10 prefixes, and 10 ASes. We disregard any bandwidth for setting up connections or other overhead but only consider the RCA or certificate data being sent over the internet. In practice, the numbers will therefore be higher.

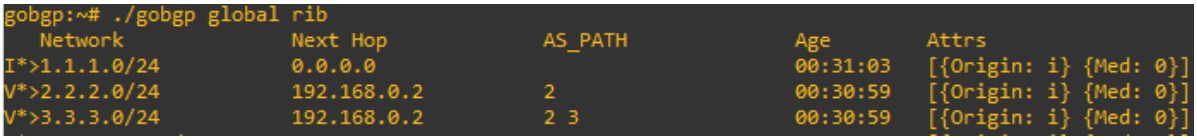
As can be seen in Table 6.4, even if the estimations on average RCA contents or average RCAs per AS are a bit off, it does show that the total amount of storage and bandwidth required is not huge, but manageable. Especially since not all ASes will use RCAs, the storage and bandwidth requirements at the beginning of deployment will be even lower. The results show that storage and bandwidth requirements are reasonable and that, in our opinion, it should not be an issue for ASes to consider RCA deployment.

6.4. Exact results

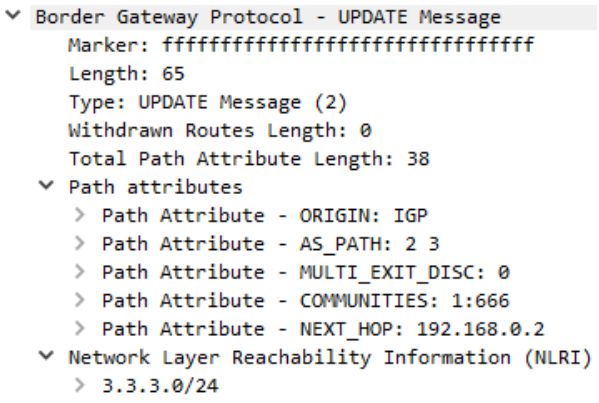
In this section, we will discuss the exact results that we collected. Each experiment is discussed independently below.

6.4.1. Community Attacks

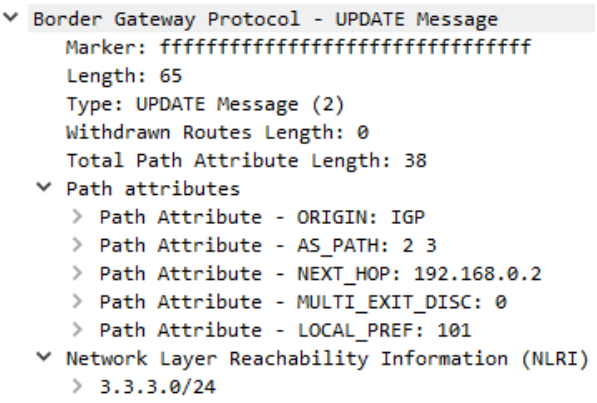
From the experiments that were run, the RCA solution successfully sends a mitigation for every community attack. Note that the RCA needs to be configured correctly for this to work. We will explain how the experiment works below, see Figure 5.1 as a reference for the setup. AS 1 has the RIB configured as seen in Figure 6.3a. After AS 2 in the architecture diagram tries to use a community value which is disallowed, in this case a blackhole attack, the RCA solution responds and mitigates the attack. The attack from AS 2 can be seen in Figure 6.3b. The RCA server from AS 1 responds by sending a mitigation, which can be seen in Figure 6.3c. The final RIB of AS 1 shows that the mitigation successfully overwrites the attack in Figure 6.3d. Note that the attack would have been successful without the mitigation since the blackhole policy sets the next hop of 3.3.3.0/24 to 192.0.2.1, which is configured as a blackhole on the router of AS 1. We repeated this experiment for every single case in the validation algorithm (see the if-statements of the validation algorithm in Chapter 5) and with different community values, including regex statements. We also repeated the experiments for other attacks different than the blackhole attack, like path prepending.



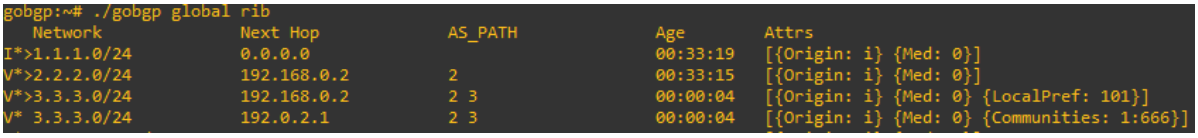
(a) The BGP table of AS 1 before the attack happens.



(b) The BGP UPDATE community attack seen on the wire from the attacker to the target using Wireshark.



(c) The BGP UPDATE mitigation seen on the wire from the RCA server to the target using Wireshark.



(d) The BGP table of AS 1 after the attack happens. Notice that the mitigation route overwrites the attacker's route. The router has configured that 192.0.2.1 is a blackhole route.

Figure 6.3: These images show the different stages of a community attack when running an RCA server. We show the BGP table before the attack happens, the BGP UPDATE message of a community attack (in this case, a simple blackhole attack for readability), the BGP UPDATE message of the mitigation, and the resulting BGP table once the mitigation has happened.

Because any community value that is not allowed causes the announcement from the attacker to be mitigated (either by dropping, cleaning, or overwriting), and assuming that the RCAs are configured correctly, any attack of the scenarios that we discussed in the background can be mitigated. We confirmed that traffic still resumes normally after mitigation (for example, traffic is not blackholed after mitigating a blackhole attack). Testing if all traffic can still reach the destination via the correct path was done in GNS3 simulations. This is a promising result as it confirms the theory of the solution that we developed.

6.4.2. Verification algorithm

The verification algorithm is measured precisely. We did this to give a good indication of the performance of individual components and show that even a laptop with adequate hardware is capable of keeping up to speed with verification. This means that expensive hardware is not needed, since a simple server can offload the routers from cryptographic operations, just like RPKI does with ROAs. The experiment works as follows:

1. Setup an RCA and embed it in an x509 certificate.
2. For this experiment it is assumed that ROA data is already verified, so we just provide the algorithm with the prefixes from the ROA.
3. Start a timer and run the algorithm. Stop the timer and get the time it took. Repeat this n amount of times (indicated per experiment in the figures).
4. After n runs, calculate the median time.
5. The verification algorithm can have three cases, namely the case where a signature is invalid, a signature is valid but the RCA is not, and a valid signature and valid RCA. All three cases can happen during verification, but the worst case in terms of time for a router would be that all signatures and RCAs are valid. Therefore, the last experiment is the most important one since that will be the greatest potential bottleneck.
6. We repeat the experiments for three different encryption schemes, namely RSA, ECDSA, and Ed25519. We chose the key sizes with a security strength of 128 according to the NIST recommendations on key sizes [41]. This means that RSA uses 3072 bits, ECDSA 256 bits, and Ed25519 also 256 bits. These runs are run with RCAs containing 10 communities and 10 prefixes, and with RCAs containing 1 community and 1 prefix.
7. We repeat the experiments for Ed25519 with an increasing number of communities and an increasing number of prefixes to see how verification time is influenced and how it scales.

First, we need to define what we mean by increasing variables. There are three variables in this experiment, and we will explain for each one what it means to change that variable:

- Encryption scheme: this is the underlying encryption scheme for signing and verifying the certificates. Changing encryption schemes can change signing and verification times, which will influence how the verification algorithm performs.
- Communities: the number of communities increases the size of the RCA and therefore the size of the certificate. Furthermore, when parsing an RCA from a certificate, we perform regex validation. We could have chosen the number of ASes as well since that also increases RCA size, but we decided to go with communities since they are also verified in terms of regex. When we talk about increasing the number of communities, we define them as follows. 1 community is defined as [”1:1”], 3 are constructed as [”1:1”, ”1:2”, ”1:3”], etc.
- Prefixes: the number of prefixes increases the RCA size, but prefixes themselves are verified additionally. In the algorithm, we check if the ROA prefixes cover the RCA prefixes, which means that more prefixes will cause more calculations. The ROA prefix is defined as 0.0.0.0/0, meaning all possible prefixes. When talking about increasing the prefixes, we construct it as follows: 1 RCA prefix is 0.0.0.0/32, 3 prefixes are 0.0.0.0/32, 0.0.0.1/32, 0.0.0.2/32, etc. You can already tell from this that it is more performant to aggregate prefixes, but we perform the experiments this way to show how a lot of prefixes can cause longer verification times. We use the 0.0.0.0/0 construction to make sure each additional prefix is covered.

Now that the variables and how they are increased are clear, we can continue with the results. The results with the three encryption schemes can be seen in Figure 6.4. In this figure, we plotted the boxplots for each encryption scheme. In Figure 6.4a you can see the experiments with 1 prefix and community, and in Figure 6.4b you can see the experiments with 10 communities and prefixes. From these results, we can see that Ed25519 performs the best in the experiments while having the same

security strength as the other schemes according to the NIST recommendations. We thus recommend using Ed25519 for RCA certificates. Note that we did not draw the outliers in the figures with boxplots. They are still present in the datasets and the other calculations, but we decided to not draw them to make the boxplots easier to visualize and interpret.

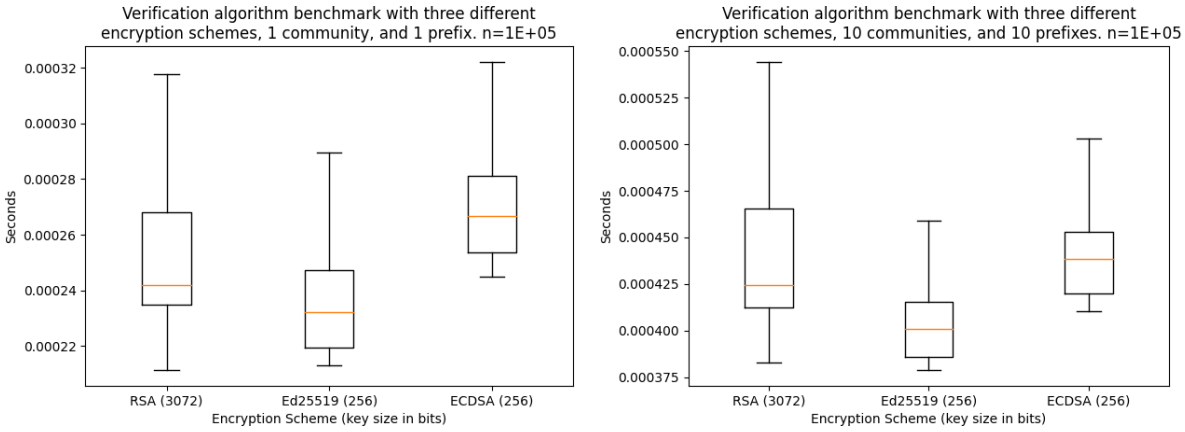


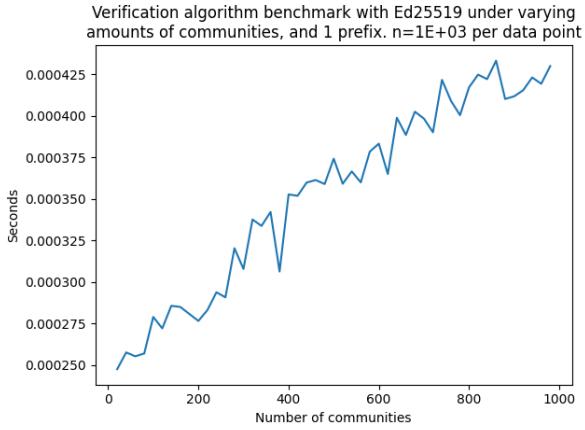
Figure 6.4: Benchmark of the verification algorithm with three different encryption schemes. We both show the experiment with 1 community and prefix, and with 10 communities and prefixes.

In Table 6.5, we show the raw numbers to make them easier to read instead of looking at the plots. From these results, we can calculate that we can successfully verify 2325 RCAs per second with 10 communities and 10 prefixes. This seems on the low side, but there should not be a continuous stream of RCAs every second, but in batches every X minutes. By multi-threading the algorithm, the number of announcements verified per second can be greatly increased. The algorithm can also be further optimized, perhaps in another language than Python. This number of RCAs per second depends on the number of communities used in an RCA, and the number of prefixes. For example, with only 1 community and prefix, we can suddenly verify 3949 RCAs per second. Next, we will look at how communities and prefixes influence the scalability of the algorithm.

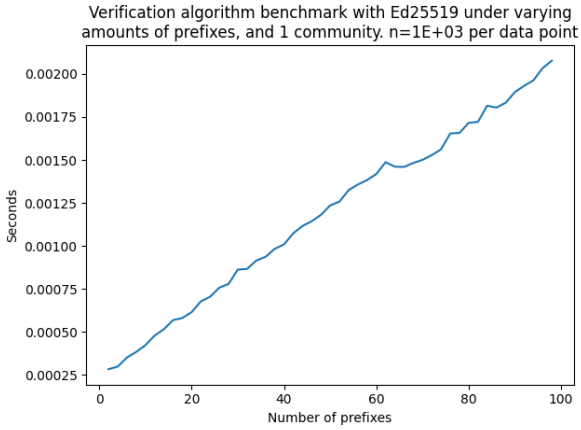
Verification result	Median
Valid signature and valid RCA, 10 communities and 10 prefixes	0.00043010000000265336 seconds
Valid signature and valid RCA, 1 community and 1 prefix	0.00025320000000306209 seconds

Table 6.5: Verification performance results with Ed25519. n=1E+05

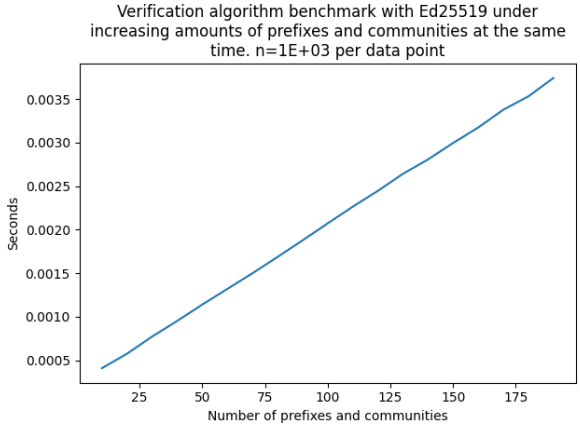
In Figure 6.5a, we can see how the verification time behaves when increasing the number of communities with Ed25519 encryption. We can see that it scales linearly, which is a positive result. In Figure 6.5b, we can see how the verification times are influenced when increasing the number of prefixes. The reason why the increase is steeper than with communities is that communities only increase the size of an RCA in the verification algorithm. No calculations other than verifying the certificate, and regex validation, are influenced by this size increase. However, the prefixes of an RCA are also verified if they are covered by a ROA prefix. The calculation for this is apparently heavy compared to only increasing the RCA size, and will thus give us a steeper increase. From the results, we can also see that it scales linearly.



(a) Varying amounts of communities.



(b) Varying amounts of prefixes.



(c) Varying amounts of both communities and prefixes at the same time.

Figure 6.5: Benchmark of the verification algorithm with both varying amounts of prefixes and communities. It uses the Ed25519 encryption scheme.

The last question to answer is the scalability of the whole algorithm. The two variables independently scale linearly, but it could turn out that the algorithm is polynomial when scaling them at the same time. In Figure 6.5c we increased both variables at the same time. From this figure, we can see that the whole algorithm scales linearly as well.

We recommend the following advice for each of the two variables that influence verification speed:

- The number of communities does not influence verification speed that much compared to prefixes. It could be slightly improved by generalizing communities via regex.
- The number of prefixes does influence verification speed significantly. Therefore, we recommend aggregating prefixes as much as possible (like /24 instead of multiple /32 prefixes). This will improve verification time quite a bit.

6.4.3. Validation algorithm

For the validation algorithm, the experiment looks a bit different:

1. Setup an RCA.
2. Create announcements for three possible validation cases that will happen in the wild: RCA not found, best-case validated, and average-case validated.
3. RCA not found is simple, the announcement is from an AS that does not have any RCAs. Every single announcement that passes a router will take at least this much longer to process an

announcement.

4. Best-case validated means that there are RCAs, but the announcement does not contain any communities, meaning that it is automatically valid.

5. Average-case validated means that we have to go through the whole validation algorithm. This time will be dependent on several variables. This experiment is the most important to see if there is a bottleneck for a router and how many announcements a router can process per second.

6. Run each of the cases n times and output the median results and plots.

7. Repeat the experiments by varying the different variables like number of prefixes, number of communities, etc.

First, we need to define what we mean by increasing variables. There are four variables in this experiment, and we will explain for each one what it means to change that variable:

- Communities: the number of communities, either in an announcement or in an RCA, means that more regex checks need to be performed to validate an announcement. When we talk about increasing the number of communities, we define them as follows. 1 community is defined as [”1:1”], 3 are constructed as [”1:1”, ”1:2”, ”1:3”], etc. When we talk about increasing the communities, we mean that both the communities in an announcement and the RCA are increased, as both have an impact on performance. Note, that we disregard complex regexes in this experiment and only use simple ones. We previously mentioned that a malicious AS could construct a regex that takes too long on purpose. Perhaps there should be restrictions on the regex besides a maximum time to combat any potential DoS attempts. However, this is disregarded for this experiment where we assume that regexes are simple.

- Prefixes: in the validation algorithm, we check if the announcement is covered by an RCA before proceeding with additional validation checks. If an announcement or RCA contains multiple prefixes, then additional calculations need to be performed. The announcement prefix is defined as 0.0.0.0/0, meaning all possible prefixes. When talking about increasing the prefixes, we construct it as follows: 1 RCA prefix is 0.0.0.0/32, 3 prefixes are 0.0.0.0/32, 0.0.0.1/32, 0.0.0.2/32, etc. You can already tell from this that it is more performant to aggregate prefixes (like /24 instead of multiple /32 prefixes), but we perform the experiments this way to show how a lot of single prefixes can cause longer validation times. We use the 0.0.0.0/0 construction to make sure that each additional prefix is covered.

- ASes: the number of ASes are the ASes that are allowed to be on the path of an announcement. So if an announcement contains ASes [2, 3], and the RCA contains ASes [2, 3, 4, 5], we perform a subset calculation to see if the announcement ASes are covered by the RCA ASes. When we talk about increasing the number of ASes, we both increase the ASes in the announcement and the RCA. They are constructed as follows: 1 AS means [1], 3 ASes mean [1, 2, 3], etc. By increasing both the announcement and RCA ASes, the subset calculation will take the maximal amount of time. Note that in practice, it will be much more likely if the subset calculations are smaller.

- RCAs: the number of RCAs, is the number of RCAs that are checked to see if an announcement is valid. If the first RCA does not cover the announcement in any of the checks, the algorithm checks the next RCA, etc. When we talk about increasing the number of RCAs, we are repeating the average-case validated scenario n times. In reality, it will be much more likely that the algorithm proceeds to the next RCA in one of the earlier checks instead of continuing to the end each time.

Now that the variables and how they are increased are clear, we can continue with the results. The results for the different validation cases are listed in Table 6.6 with both 1 of each variable, and 10 of each variable. The boxplots for these results can be seen in Figure 6.6.

In Figure 6.7, we have plotted the algorithm against increasing amounts of each variable, and we can see in the plot how the algorithm time increases and behaves depending on which variable is increased.

Validation result	Median
RCA not found 1 community, 1 prefix, 1 AS, and 1 RCA	0.00000040000000001150 seconds
RCA not found 10 communities, 10 prefixes, 10 ASes, and 10 RCAs	0.00000040000000001150 seconds
Best-case validated 1 community, 1 prefix, 1 AS, and 1 RCA	0.00000060000000035032 seconds
Best-case validated 10 communities, 10 prefixes, 10 ASes, and 10 RCAs	0.00000060000000012828 seconds
Average-case validated 1 community, 1 prefix, 1 AS, and 1 RCA	0.00002250000000003638 seconds
Average-case validated 10 communities, 10 prefixes, 10 ASes, and 10 RCAs	0.00102210000000013679 seconds

Table 6.6: Validation performance results with n=1e5.

From the previously mentioned results and figures, we can already draw some conclusions. First, we can see that the algorithm has excellent performance for low amounts of variables. However, as we increase each variable, we can see that it gets significantly slower. For each variable, we can draw the following conclusions and recommendations:

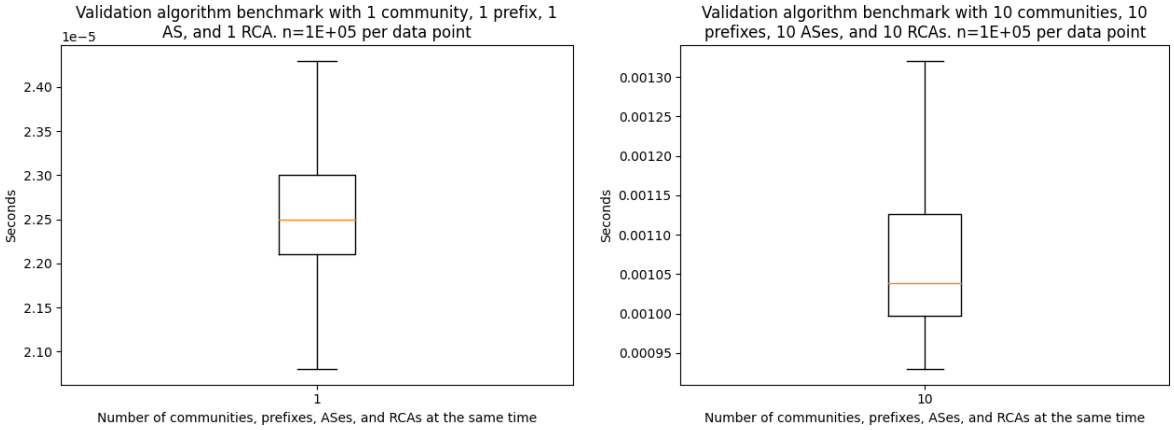
- Increasing the communities does not influence the timing that much. While keeping all variables at 1, instead of processing 44444 announcements per second with 1 community, we can process about 10000 per second with 200 communities. As we will see later, this is still a reasonable number. If many communities need to be specified, we recommend generalizing communities with regexes, on the condition that the generalized regex is quicker than individually specifying the communities.

- When looking at prefixes, we can see that it significantly influences validation speed. As we also have seen in the verification algorithm, calculating if prefixes cover other prefixes is an expensive operation. To improve validation time, we recommend aggregating prefixes. As an example, instead of specifying several /32 prefixes, validation speed can be improved by specifying a single /24 prefix. This is also a common practice for BGP itself.

- Number of ASes influences the validation speed the least. There is not a specific recommendation here.

- Number of RCAs that need to be checked before validation succeeds has the biggest impact. This makes sense, as in this experiment the algorithm repeats all validation checks for the number of RCAs that was specified. In the real world, this will be faster, as validation can already move to the next RCA if a condition is found early on that shows that the RCA does not cover the announcement. Generalizing RCAs can improve validation speed. As an example, instead of creating an RCA that covers prefix p with community value 1, and another RCA that covers prefix p with community value 2, it is quicker to make 1 RCA for p containing both community values.

Using the results from Table 6.6, we can calculate statistics like the number of announcements per second, and total processing time when validating an entire full BGP table. These statistics can be seen in Table 6.7. In the simulations performed, when using full BGP tables, we see that the router



(a) 1 of each variable.

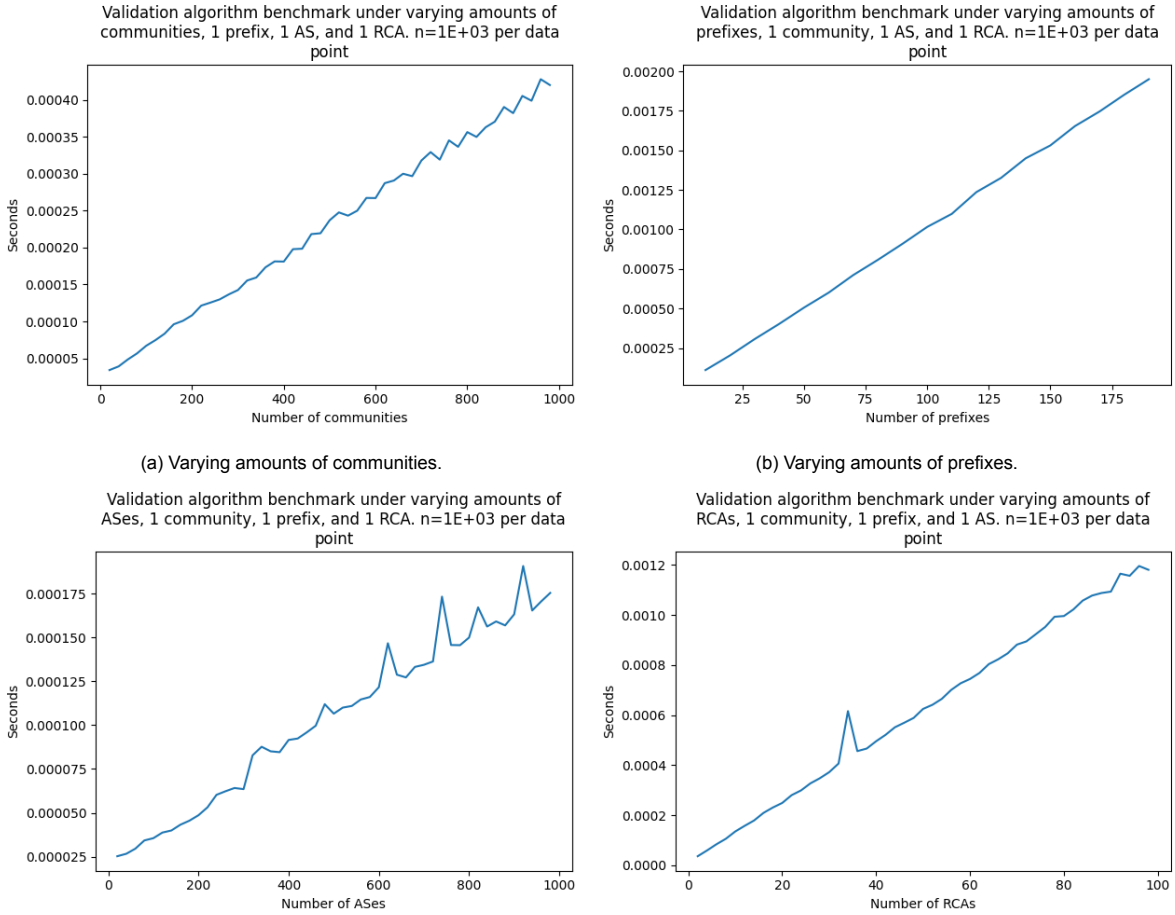
(b) 10 of each variable.

Figure 6.6: Boxplots of the validation benchmark with 1 and 10 of each variable.

sends between 70k and 80k announcements every 10 seconds to the validation algorithm when injecting a full BGP table. Receiving BGP tables also happens when a router is starting and receives the BGP tables from its peers. However, we do not know how many announcements are sent when this happens on the internet, and if the amount is equivalent to the 70k in 10 seconds we saw during experiments. The algorithm can keep up with this rate when we use 1 for each variable. However, with 10 for each variable, the validation becomes a bottleneck during startup. If we use more realistic numbers however, and keep the advice for variables in mind, we can calculate the validation speed for an RCA with 10 communities, 3 prefixes, 10 ASes, and 4 RCAs. With n=1e5, we get a median speed of 0.00024520000000194386 seconds, meaning that it processes 4078 announcements per second, and 40783 every 10 seconds. Keep in mind that this is single-threaded and that not every announcement goes through complete validation in reality, but it does in our experiments due to the setup since we assume each announcement takes the average-case validated. There will be higher percentage of announcements that result in RCA not found or a best-case validated compared to average-case validated when running on the internet, since not all ASes will deploy RCA, especially in the beginning. This means that in a realistic scenario, and with multi-threading, the solution should be fast enough to keep up to speed when a router starts and receives full tables from peers. We also know from Table 6.2 that once a router is running and is stable, the number of updates per day is relatively low compared to starting a router and receiving a full BGP table. Since there can be peaks as high as 1 million IPv4 announcements per day for a router, our solution can easily keep up with this rate. Therefore, even if we assume the performance of 978 announcements per second (single-threaded, average-case validated for each announcement), there will not be a bottleneck when considering the rate of announcements seen on the internet. This is because at this rate of 978 per second, we can process 86400 (number of seconds in a day) * 978 = 84.499.200 announcements per day. This means that our algorithm will potentially only result in longer startup times. We believe that increased security is worth the extra time when starting a router, as long as it does not bottleneck the performance once running stable on the internet.

In the beginning when ASes rollout RCA, it will not take longer than a few seconds at maximum to validate an entire BGP table. However, if the entire internet decides to use RCA (which is unlikely, since RPKI usage is not even close to that [42]), startup times will take longer if the code is not optimized further and if there are no further hardware improvements. ASes need to decide if the additional time is worth the extra security that it brings. Do note however, that this experiment assumes that validation is not multi-threaded. Multi-threading the validation code will improve the performance, and if the solution is run in BMP monitoring mode, then the router itself will have no additional load. This is because in that case, the validation is done on the server-side, at the cost of a small increase in mitigation time.

Finally, from Figure 6.8, we can see how the validation time is influenced when varying all variables of the validation algorithm at the same time. We have already seen in the figures that increasing each



(a) Varying amounts of communities.

(b) Varying amounts of prefixes.

(c) Varying amounts of ASes.

(d) Varying amounts of RCAs.

Figure 6.7: Benchmark of the validation algorithm with varying amounts of communities, prefixes, ASes, and RCAs.

variable independently scales linearly. However, when increasing them at the same time, we see in the figure that the algorithm has polynomial scaling. This makes sense when looking at the code for the validation algorithm. Thus, polynomial scaling is expected and not necessarily a bad result. When running the algorithm with realistic numbers, and even without optimisations like multi-threading, it is performant and seems to keep up with the rate of announcements seen on the internet.

6.5. Threat model

As we have seen in the experiments, there is a small time window (~0.12 seconds) where the community attack is considered successful before the mitigation is effective when using BMP. To demonstrate the capabilities of an attacker, and the damage that can be done in 0.12 seconds, we will construct a threat model and analyze what damage can be done. Please note that this threat model is only applicable when RCAs are used via BMP. When RCAs are implemented on a router itself, then an attack can be immediately stopped by dropping or cleaning an announcement (unless an AS misconfigured their RCAs).

We will construct a threat model and analyze the damage for the three kinds of community attacks that we have seen in the background chapter. A realistic threat actor is described as follows:

- The threat actor can control an AS. This AS can send, modify, and drop arbitrary BGP messages.
- We assume that the attacker AS has rich connectivity with peers, meaning that it can launch a wide range of community attacks against different targets. All three attacks described in the background can be launched by the threat actor.

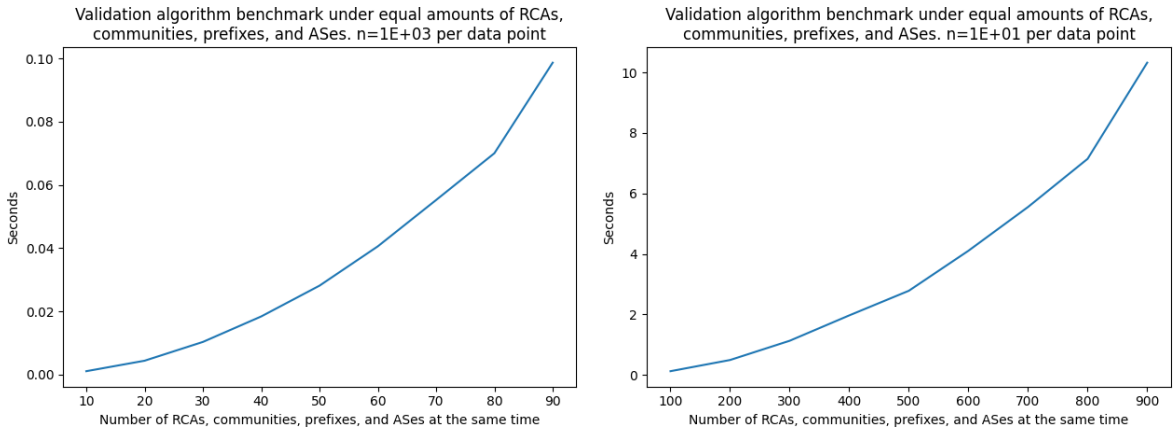
Validation result	Announcements per second	Processing time full BGP table
RCA not found 1 for each variable	2499999	0.37861400001 seconds
RCA not found 10 for each variable	2499999	0.37861400001 seconds
Best-case validated 1 for each variable	1666666	0.56792100033 seconds
Best-case validated 10 for each variable	1666666	0.56792100012 seconds
Average-case validated 1 for each variable	44444	21.2970375 seconds
Average-case validated 10 for each variable	978	967.4534235 seconds

Table 6.7: Validation performance results in terms of announcements per second and processing time.

- The neighbours of the attacker AS deploy RCA and RPKI, since the attack will always be successful otherwise. We are considering the damage that can be done when RCA protection is used.
- The threat actor does not have unlimited bandwidth or unlimited compute power, as this is not a realistic scenario.
- The victims and targets of the threat actor configured their RCAs correctly and are vulnerable to all community attacks when not using RCAs (meaning that they offer services with path-prepending communities, blackhole communities, etc.).
- The targets deploy RCA using BMP, so the time to mitigation is $\sim$0.12 seconds per attack.
- The threat actor only uses attacks that exploit community values. BGP hijacks or other BGP-related attacks are not considered in the threat model.

With the above description of a realistic threat actor, we can take a look at the three different community attacks and analyze the damage of $\sim$0.12 seconds where the attack is not mitigated yet.

- Blackhole attack: When the attacker has launched a successful blackhole attack, traffic to the destination IP that is targeted is unreachable for 0.12 seconds for traffic that goes through the target AS to the destination (or from the destination through the target AS). Depending on the connectivity with peers of the victim AS, this can either drop lots of traffic or only a small portion. Afterwards, the mitigation kicks in and the destination is reachable again through the target AS. There are two scenarios for the attacker here:
    1. The attacker only launches 1 attack. In this case, TCP traffic will recover since most connections will close after an idle time in the order of minutes. Clients will notice a small delay but immediately recover after it. In the case of UDP, traffic is lost and the damage depends. Usually, UDP is used when traffic is allowed to fail anyways for additional performance (like with video streaming), so we should not consider the payload of UDP valuable here. TCP should have been used otherwise. Clients will notice that the traffic was dropped depending on the application that uses the data. In the case of video streaming, the frame will hang for a short amount of time before resuming.



(a) Variables until 100.

(b) Variables until 1000.

Figure 6.8: Benchmark of the validation algorithm while increasing all variables at the same time.

2. The attacker launches consecutive attacks. In both cases of UDP and TCP, the impact is as big as a successful blackhole attack. Either there is very low throughput through the target AS to the destination, or the connection becomes completely unusable. We consider both of these cases as too much of an impact and comparable to a successful blackhole attack. However, reviewing the logs of BGP or the RCA security solution will immediately reveal such an attack going on. Administrators can take action based on this (for example by disallowing the community). An alerting mechanism can also be implemented that detects such consecutive attacks and notify administrators since these consecutive attacks are easy to notice.

- Traffic steering: The scenario is the same as described above. However, in this case, the traffic is lost for a brief moment due to a route change. This can cause issues to reach the destination properly through the target AS. If that happens once, this might not be a huge issue for the end-user. If it happens continuously, then there will be a noticeable increase in latency or it can even cause reachability issues. Depending on the application, this can either have a big impact or not. Note that continuous route switching, also called route flapping, leads to a higher router processing load caused by instability and prevents the network from converging. This will cause reachability issues. Continuous traffic steering can thus make the destination unreachable through the target AS, while a one-time traffic steer is only successful for ~0.12 seconds. In these 0.12 seconds, the attacker can eavesdrop on the communication if we assume that BGP converges immediately. However, in practice, convergence will take a while (see Table 6.2), and the mitigation already takes place after 0.12 seconds. So if it happens once, the traffic is probably never successfully steered in practice on the internet and the destination is just unreachable for a bit for traffic that goes through the target AS to the victim (or from the victim through the target).
- Route manipulation: Just like before, a successful route manipulation attack will cause certain peers of an IXP to not receive certain prefix advertisements. If it happens once, a peer will not receive the prefix advertisement for ~0.12 seconds. Depending on the reachability of this peer, either traffic is routed via a different route or the peer does not have reachability to the prefix for a brief moment. Continuous route manipulation can also cause a situation where route flapping occurs if we assume that the route advertised to the peer is preferred over the other routes, which will cause a route change. Otherwise, we expect that there will be a slightly higher load on the router but probably no impact on reachability.

From above, we can see that in practice the threat actor can only cause damage by disturbing the reachability through the target AS to the destination, either shortly if the threat actor attacks once, or continuously if the attacker keeps repeating the attack. The damage depends on if the victim AS is still reachable through other peers and what is running or making use of the destination. As an example, in the case of stock trading, there can be an economic impact if the stock trade exchange becomes

unreachable. The damage of these kinds of attacks needs to be evaluated similarly to DoS and DDoS attacks.

7

Discussion

In this chapter, we discuss the results of this thesis that were gathered in Chapter 6 and the solution that was designed itself. Furthermore, we will discuss the limitations of the research that was done and provide directions for future research. Afterward, we will answer the subquestions of this research as stated in the methodology.

As discussed and motivated in Chapter 5, the final solution design that protects the BGP community attribute provides a tradeoff between the different metrics that a BGP security solution should ideally have, as discussed in Chapter 2. We protect the community value by authorizing their usage through a new RPKI object called Route Community Authorization (RCA). The results of the experiments as seen in Chapter 6 look promising. We first showed that community attacks can be stopped using correctly configured RCAs by either dropping or cleaning an announcement with invalid communities. Furthermore, we predicted both with a theoretical analysis and with a combination of benchmarks in simulation software that the solution should be able to keep up with the rate of updates that are sent on the actual internet. The only place where we identified that it could potentially cause delays is during router startup where full BGP tables are received in a short amount of time. However, we argued that this delay will probably not happen in practice based on deployment rates when looking at RPKI and with some performance improvements we suggested for RCAs. If it does happen, we think that longer startup times can be acceptable (if it is not too long of course) for the added security benefits of RCAs. The benchmarks show that the verification algorithm has linear scaling and the validation algorithm polynomial scaling. We also made suggestions for how ASes can configure RCAs to make sure it is performant and how performance can be improved in the future, for example, by multi-threading the algorithms. Our predictions, like scalability to the size of the internet, that we made need to be confirmed using additional experiments on the internet in future research.

The security of participating ASes is determined by the configuration of RCAs, and how many ASes are participating. We have seen in the results that either the target of the attacker, or an AS on the path between the attacker and the target, needs to deploy RCA to successfully mitigate an attack, on the condition that RCAs are configured correctly. We did discover that in certain cases where RCAs rely on the ASes and AS-PATH length attribute, that these RCAs can be circumvented if an attacker modifies the AS-PATH attribute in the BGP UPDATE message. Since RCA cannot provide integrity protection of the AS-PATH attribute in BGP, we recommend deploying ASPA in the future when it becomes available to limit this risk. Furthermore, we made recommendations to limit execution time for regexes, or more strict requirements for them, to limit any ASes from potentially configuring regexes that take too long to execute.

The idea of RCAs, and ROAs, is that you are protecting other participants, but that also means that there is (almost) no security when deployment is low. This also causes an issue for incentive, as you want to protect yourself when choosing a security solution, not essentially others. Deployment of the solution by ASes will be the biggest obstacle for the future of RCA. Since incentive is a big issue for deployment of security solutions of BGP in general, it can be generated if providers are enforced to

deploy security solutions by law for example, or if larger ASes implement security proposals first.

Since router hardware currently deployed is not capable of performing cryptographic operations per route announcement, this means that certain security goals like the integrity of the community attribute cannot be achieved. Using RPKI objects, we can offload cryptographic operations from the router to a verification server and offer authorized use of community values. This approach provides a solution that can be used with current hardware routers (currently through BMP). But that also means that when RCAs are configured too broad, attackers can still abuse them. Due to the current hardware, we think that the approach of RCA is the best way to protect against community attacks at the moment since it does not limit the legitimate usage of community values like other mitigations do. Only an additional server is needed, without too much computational power since experiments were done on just a laptop with adequate hardware in our case. Furthermore, since communication is done out of band, BGP itself does not need to be altered. In the future though, we suggest using BGPsec and extending it with community values as well for better security when routers are capable of running BGPsec. Using a BGPsec approach will improve security and provide integrity of the community attribute, which are security properties that RCA cannot provide.

There are some limitations of the research due to a limited time frame and the scope of this thesis, as explained in Chapter 4. which means that certain goals were not achievable. Additional experiments are needed to see if RCAs can successfully capture the RCA usage of ASes, if these RCAs are still performant enough (both on software and real hardware routers), and if it scales successfully to the size of the internet. We currently used our parameters for RCA configuration, but it could be that ASes have different needs than we predicted, which could impact the performance. Furthermore, due to the scope and time frame of the thesis, we were only able to perform the experiments in software simulations, where ideally they are performed on actual hardware routers.

Other future work can be directed at implementing router support for RCAs and repeating the experiments on those routers. Then the actual impact on routers can be measured, and there is no time to mitigation window anymore as there is with BMP. This improves the security of RCAs. Furthermore, future work can also be directed at adjusting the RCA model. For example, RCA can be slightly adjusted to just notify administrators of violations instead of performing automatic mitigation. In addition, the model of RCAs could be improved with additional attributes to enhance the capabilities, or a more efficient algorithm that we did not think about. With input from other researchers or the community, RCA can be further improved. Also, as mentioned in the scope, we only worked with the normal community attribute and IPv4. The RCA model can be extended to also include the large and extended community attribute, and the same goes for IPv6 support. Additionally, in the scope of this thesis, it was defined that the semantics of the community values are not verified or validated by an RCA, only the values that are defined by the origin AS. It would be an exciting direction, and quite the challenge, to research a method to verify and validate the semantics of community values. This can strengthen the security guarantees that RCA can offer. Last, but not least, we encourage researchers to perform additional research on mitigating community attacks. This could result in a different solution compared to RCAs that use different tradeoffs to protect against community attacks.

Now that we have discussed the results of the thesis, the limitations, and the future work, we can answer the subquestions of the thesis.

What has already been done to secure BGP against community attribute attacks?

The answer to this research question is the result of Chapters 2 and 3, where we discussed the background of both BGP, the community attribute, BGP security and attacks, notable security solutions, community attacks and their (not so effective) mitigations, and related work in this research area. Using this background that we acquired, we were able to use the literature to come up with a new security solution. We used metrics that are important to consider, researched why certain security solutions failed and why others (partly) succeeded, what is important for ASes to consider deploying security solutions, etc. Based on this information we can estimate and try to arrive at a solution that makes use of history by using what succeeded, and avoiding what failed.

What is important to consider a solution viable for deployment, and with which metrics can we measure that?

As we briefly mentioned in the previous question, metrics are important to define beforehand so we know what we need to keep in mind when developing a solution, and with which metrics we can assess the effectiveness. In summary, we need to assess security solutions based on four different properties, namely:

- Security properties: What kind of security does the solution provide, does it introduce new vulnerabilities, which vulnerabilities remain in BGP with this solution, and how many ASes need to deploy it before security benefits arise?
- Performance properties: Does the security solution influence the stability on the internet, cause an increase in convergence, does it scale, and how much overhead does it introduce in terms of computational, bandwidth, and storage overhead?
- Privacy properties: Do ASes need to expose their routing policies, business relationships, or other private information to make use of the security solution?
- Deployability properties: Is the solution easy to deploy, can it be deployed incrementally, is additional hardware needed, and is there interoperability with legacy BGP?

Of course, one solution cannot excel in all of these properties, tradeoffs need to be made. Which tradeoffs are made is important for a solution to be considered viable for deployment, and we can use the knowledge from the previous question to estimate what an ideal tradeoff would be. The result of how our solution scores at these metrics is presented in Chapter 6.

How can we meaningfully develop and test solutions without deploying them live on the internet?

This is an important question, but at the same time difficult to answer. One option would be to simulate the routers and create different scenarios in the simulation software to test properties like scalability. However, to be able to measure those kinds of metrics, lots of computational power is needed, as simulations are computationally and memory-heavy tasks. The best solution is still to be able to run experiments on the internet using the PEERING testbed for example. However, in the end, we used a combination of simulation scenarios, benchmarks, and theoretical analyses to be able to assess and test our security solution. We created several scenarios with a handful of routers to test if the solution even works. Using benchmarks, we estimated how much overhead there is when processing announcements, how much storage is required, etc. These results are presented in Chapter 6. As indicated in future work, more research is needed to confirm our predictions about scalability by testing the solution on hardware routers and on the internet.

How can we verify the community attribute while satisfying the requirements for a viable solution?

Using the information from the first subquestion, we decided to use RPKI to create a new object where an AS authorizes which community values can be used. These objects can be cryptographically verified and are called RCAs. The construction has several benefits and drawbacks, which we discussed in this thesis and in this chapter. We think that this approach using RPKI is currently the best since current routers deployed are not capable of performing cryptographic operations on every announcement. With the RPKI construction of RCAs, cryptographic operations are offloaded to a separate server that verifies the certificates instead of putting this load on the router. The verified RCAs can then be sent to the router where the router can validate each announcement using the RCAs. In the future, we suggest using BGPsec and extending it with community values for better security. Using a BGPsec approach will improve security and provide integrity of the community attribute, which is something that RCA cannot provide. Until routers that are deployed on the internet can handle BGPsec, we believe that this is the best solution in the meantime to mitigate community attacks compared to the other mitigations.

8

Conclusion

In this thesis, we have developed a security solution to protect against a new attack vector of BGP that makes use of the community attribute. There are no effective mitigations against these attacks to date, and none of the security extensions that have been developed for BGP so far protect the community attribute. That is why this thesis is relevant, as we have developed the first solution that provides authorized use of communities that can be cryptographically verified. The main research question for this thesis to answer is defined as follows: **How can we secure BGP community attributes against attacks?** We answered this question by creating a new RPKI object that includes information like which community values are allowed to be used and the prefixes that it covers, to name a few examples. This information restricts the attack surface of communities, but still allows for the legitimate usage of them. If ASes configure their objects correctly, other ASes can enforce the rules specified in the new RPKI object that we called Route Community Authorization (RCA) and take action when encountering announcements with disallowed communities. Our approach does not alter the BGP protocol itself, as the communication for our solution is performed out of band.

From the results, we see that the solution can successfully mitigate community attacks with properly configured RCAs. We have also seen in benchmarks that the developed solution should be able to keep up with the rate of announcements that are seen by routers on the internet. We also predict that it should be scalable to the size of the internet using a theoretical evaluation and with benchmarks. However, as mentioned in the thesis, our research was limited by performing the experiments in simulation software only. That means that we were not able to perform experiments on hardware routers or on the internet to test important properties like scalability. It is important that experiments are also performed by ASes themselves, to see if the RCAs can successfully capture their community usage and if these RCAs are still performant enough. These limitations are discussed in more detail in the discussion, together with the future work that is possible. The biggest issue that will remain for the developed solution is the initial adoption and deployment by ASes. This is because the security benefits are low in the beginning, but increase as more ASes participate. This causes a lack of incentive to initially adopt the new security solution.

As current routers used by ASes are not capable of performing cryptographic operations per announcement, we used the approach with RPKI objects to offload the cryptographic operations to a verification server. This server can forward the verified RCAs to the actual router, which means that the router does not have to perform these cryptographic operations itself. We believe that our developed solution is currently the most suited for mitigating community attacks, as it does not alter BGP itself or limit the community usage of ASes, as compared to other mitigations. In the future, when routers can handle the cryptographic load per announcement, we recommend using BGPsec and extending it to include the community attribute for better security than our solution with RPKI objects can offer, like integrity protection.

Bibliography

[1] Y. Rekhter, S. Hares, and T. Li, *A Border Gateway Protocol 4 (BGP-4)*, RFC 4271, Jan. 2006. DOI: 10.17487/RFC4271. [Online]. Available: https://www.rfc-editor.org/info/rfc4271.

[2] S. L. Murphy, *BGP Security Vulnerabilities Analysis*, RFC 4272, Jan. 2006. DOI: 10.17487/RFC4272. [Online]. Available: https://www.rfc-editor.org/info/rfc4272.

[3] A. Mitseva, A. Panchenko, and T. Engel, "The state of affairs in bgp security: A survey of attacks and defenses," *Computer Communications*, vol. 124, pp. 45–60, 2018, ISSN: 0140-3664. DOI: https://doi.org/10.1016/j.comcom.2018.04.013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S014036641731068X.

[4] C. Testart, "Reviewing a historical internet vulnerability: Why isn't bgp more secure and what can we do about it?," TPRC, 2018.

[5] R. NCC, "Youtube hijacking: A ripe ncc ris case study," Mar. 2008. [Online]. Available: https://www.ripe.net/publications/news/industry-developments/youtube-hijacking-a-ripe-ncc-ris-case-study.

[6] S. Nichols, "Aws dns network hijack turns myetherwallet into thievesetherwallet," *The Register*, Apr. 2018. [Online]. Available: https://www.theregister.com/2018/04/24/myetherwallet_dns_hijack/.

[7] D. Madory, "Large european routing leak sends traffic through china telecom," *APNIC*, Jun. 2019. [Online]. Available: https://blog.apnic.net/2019/06/07/large-european-routing-leak-sends-traffic-through-china-telecom/.

[8] F. Streibelt *et al.*, "Bgp communities: Even more worms in the routing can," in *Proceedings of the Internet Measurement Conference 2018*, ser. IMC '18, Boston, MA, USA: Association for Computing Machinery, 2018, pp. 279–292, ISBN: 9781450356190. DOI: 10.1145/3278532.3278557. [Online]. Available: https://doi.org/10.1145/3278532.3278557.

[9] Doug Madory, "Bgp/dns hijacks target payment systems," *Oracle Dyn*, Aug. 2018. [Online]. Available: %7Bhttps://web.archive.org/web/20190611040109/https://dyn.com/blog/bgp-dns-hijacks-target-payment-systems/%7D.

[10] H. Birge-Lee, L. Wang, J. Rexford, and P. Mittal, "Sico: Surgical interception attacks by manipulating bgp communities," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '19, London, United Kingdom: Association for Computing Machinery, 2019, pp. 431–448, ISBN: 9781450367479. DOI: 10.1145/3319535.3363197. [Online]. Available: https://doi.org/10.1145/3319535.3363197.

[11] L. Miller and C. Pelsser, "A taxonomy of attacks using bgp blackholing," in *Computer Security – ESORICS 2019*, K. Sako, S. Schneider, and P. Y. A. Ryan, Eds., Cham: Springer International Publishing, 2019, pp. 107–127, ISBN: 978-3-030-29959-0.

[12] M. Lepinski and S. Kent, *An Infrastructure to Support Secure Internet Routing*, RFC 6480, Feb. 2012. DOI: 10.17487/RFC6480. [Online]. Available: https://www.rfc-editor.org/info/rfc6480.

[13] IANA, *Autonomous system (as) numbers*, Dec. 2021. [Online]. Available: https://www.iana.org/assignments/as-numbers/as-numbers.xhtml.

[14] Cloudflare, "What is an internet exchange point?," [Online]. Available: https://www.cloudflare.com/learning/cdn/glossary/internet-exchange-point-ixp/.

[15] T. Li, R. Chandra, and P. S. Traina, *BGP Communities Attribute*, RFC 1997, Aug. 1996. DOI: 10.17487/RFC1997. [Online]. Available: https://www.rfc-editor.org/info/rfc1997.

[16] T. King, C. Dietzel, J. Snijders, G. Döring, and G. Hankins, *BLACKHOLE Community*, RFC 7999, Oct. 2016. DOI: 10.17487/RFC7999. [Online]. Available: https://rfc-editor.org/rfc/rfc7999.txt.

[17] J. Heitz, J. Snijders, K. Patel, I. Bagdonas, and N. Hilliard, *BGP Large Communities Attribute*, RFC 8092, Feb. 2017. DOI: 10.17487/RFC8092. [Online]. Available: https://www.rfc-editor.org/info/rfc8092.

[18] T. Krenc, R. Beverly, and G. Smaragdakis, "Keep your communities clean: Exploring the routing message impact of bgp communities," in *Proceedings of the 16th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '20, Barcelona, Spain: Association for Computing Machinery, 2020, pp. 443–450, ISBN: 9781450379489. DOI: 10.1145/3386367.3432731. [Online]. Available: https://doi.org/10.1145/3386367.3432731.

[19] ——, "As-level bgp community usage classification," in *Proceedings of the 21st ACM Internet Measurement Conference*, ser. IMC '21, Virtual Event: Association for Computing Machinery, 2021, pp. 577–592, ISBN: 9781450391290. DOI: 10.1145/3487552.3487865. [Online]. Available: https://doi.org/10.1145/3487552.3487865.

[20] C. Testart and D. D. Clark, "A data-driven approach to understanding the state of internet routing security," *Available at SSRN 3750155*, 2020.

[21] P. Sermpezis, V. Kotronis, A. Dainotti, and X. Dimitropoulos, "A survey among network operators on bgp prefix hijacking," *SIGCOMM Comput. Commun. Rev.*, vol. 48, no. 1, pp. 64–69, Apr. 2018, ISSN: 0146-4833. DOI: 10.1145/3211852.3211862. [Online]. Available: https://doi.org/10.1145/3211852.3211862.

[22] G. Huston, *Cidr report for 20 may 22*, May 2022. [Online]. Available: https://www.cidr-report.org/as2.0/.

[23] J. Durand, I. Pepelnjak, and G. Döring, *BGP Operations and Security*, RFC 7454, Feb. 2015. DOI: 10.17487/RFC7454. [Online]. Available: https://rfc-editor.org/rfc/rfc7454.txt.

[24] T. McDaniel, J. M. Smith, and M. Schuchard, *Flexsealing bgp against route leaks: Peerlock active measurement and analysis*, 2020. arXiv: 2006.06576 [cs.NI].

[25] N. Labs, "Securing bgp," [Online]. Available: https://rpki.readthedocs.io/en/latest/rpki/securing-bgp.html.

[26] C. Testart, P. Richter, A. King, A. Dainotti, and D. Clark, "To filter or not to filter: Measuring the benefits of registering in the rpki today," in *Passive and Active Measurement*, A. Sperotto, A. Dainotti, and B. Stiller, Eds., Cham: Springer International Publishing, 2020, pp. 71–87, ISBN: 978-3-030-44081-7.

[27] ARIN, *Route origin authorizations (roas)*, Jun. 2022. [Online]. Available: https://www.arin.net/resources/manage/rpki/roa_request/.

[28] Y. Gilad, A. Cohen, A. Herzberg, M. Schapira, and H. Shulman, "Are we there yet? on rpki's deployment and security," Feb. 2017. DOI: 10.14722/ndss.2017.23123.

[29] M. Lepinski and K. Sriram, *BGPsec Protocol Specification*, RFC 8205, Sep. 2017. DOI: 10.17487/RFC8205. [Online]. Available: https://www.rfc-editor.org/info/rfc8205.

[30] A. Azimov, E. Bogomazov, R. Bush, K. Patel, and J. Snijders, "Verification of AS_PATH Using the Resource Certificate Public Key Infrastructure and Autonomous System Provider Authorization," Internet Engineering Task Force, Internet-Draft draft-ietf-sidrops-aspa-verification-08, Aug. 2021, Work in Progress, 15 pp. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-sidrops-aspa-verification-08.

[31] C. Testart, P. Richter, A. King, A. Dainotti, and D. Clark, "Profiling bgp serial hijackers: Capturing persistent misbehavior in the global routing table," in *Proceedings of the Internet Measurement Conference*, ser. IMC '19, Amsterdam, Netherlands: Association for Computing Machinery, 2019, pp. 420–434, ISBN: 9781450369480. DOI: 10.1145/3355369.3355581. [Online]. Available: https://doi.org/10.1145/3355369.3355581.

[32] M. Nawrocki, J. Blendin, C. Dietzel, T. C. Schmidt, and M. Wählisch, “Down the black hole: Dismantling operational practices of bgp blackholing at ixps,” in *Proceedings of the Internet Measurement Conference*, ser. IMC ’19, Amsterdam, Netherlands: Association for Computing Machinery, 2019, pp. 435–448, ISBN: 9781450369480. DOI: 10.1145/3355369.3355593. [Online]. Available: https://doi.org/10.1145/3355369.3355593.

[33] T. Hlavacek *et al.*, “Disco: Sidestepping rpki’s deployment barriers,” *Network and Distributed System Security Symposium (NDSS)*, DOI: 10.14722/ndss.2020.24355. [Online]. Available: https://par.nsf.gov/biblio/10155040.

[34] B. Schlinker, T. Arnold, I. Cunha, and E. Katz-Bassett, “Peering: Virtualizing bgp at the edge for research,” in *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies*, ser. CoNEXT ’19, Orlando, Florida: Association for Computing Machinery, 2019, pp. 51–67, ISBN: 9781450369985. DOI: 10.1145/3359989.3365414. [Online]. Available: https://doi.org/10.1145/3359989.3365414.

[35] O. Borchert, K. Lee, K. Sriram, D. Montgomery, P. Gleichmann, and M. Adalier, “Bgp secure routing extension (bgp-srx): Reference implementation and test tools for emerging bgp security standards,” National Institute of Standards and Technology, Tech. Rep., 2021.

[36] A. Cohen, Y. Gilad, A. Herzberg, and M. Schapira, “Jumpstarting bgp security with path-end validation,” in *Proceedings of the 2016 ACM SIGCOMM Conference*, ser. SIGCOMM ’16, Florianopolis, Brazil: Association for Computing Machinery, 2016, pp. 342–355, ISBN: 9781450341936. DOI: 10.1145/2934872.2934883. [Online]. Available: https://doi.org/10.1145/2934872.2934883.

[37] D. Freedman *et al.*, *Manrs implementation guide*, Jan. 2017. [Online]. Available: https://www.manrs.org/isps/bcop/.

[38] J. Scudder, R. Fernando, and S. Stuart, *BGP Monitoring Protocol (BMP)*, RFC 7854, Jun. 2016. DOI: 10.17487/RFC7854. [Online]. Available: https://www.rfc-editor.org/info/rfc7854.

[39] G. Huston, *As65000 bgp routing table analysis report*, May 2022. [Online]. Available: https://bgp.potaroo.net/as2.0/bgp-active.html.

[40] ——, *Bgp in 2021 – bgp updates*, Jan. 2022. [Online]. Available: https://labs.apnic.net/?p=1559.

[41] NIST, *Nist report on cryptographic key length and cryptoperiod*, May 2020. [Online]. Available: https://www.keylength.com/en/4/.

[42] ——, *Nist rpki monitor*, May 2022. [Online]. Available: https://rpki-monitor.antd.nist.gov/.