BlockJack: Towards Improved Prevention of IP Prefix Hijacking Attacks in Inter-domain Routing via Blockchain

I. Wayan Budi Sentana[a], Muhammad Ikram[b] and Mohamed Ali Kaafar[c]
Department of Computing, Macquarie University, 4 Research Park Drive, Macquarie Park, Sydney, Australia

Keywords: Blockchain, Border Gateway Protocol, Prefix Authorization, Prefix Hijacking, Prefix Verification.

Abstract: We propose “BlockJack”, a system based on a distributed and tamper-proof consortium Blockchain that aims at blocking IP prefix hijacking in the Border Gateway Protocol (BGP). BlockJack provides a synchronization amongst a BlockChain and BGP networks through interfaces ensuring operational independence. This approach preserves the legacy system and accommodates the impact of a race condition if the Blockchain process exceeds the BGP update intervals. BlockJack is also resilient to dynamic routing path changes during the occurrence of the IP prefix hijacking in the routing tables. We implement BlockJack using `Hyperledger Fabric Blockchain` and `Quagga` software package and we perform an initial set of experiments to evaluate its efficacy. We evaluate the performance and resilience of BlockJack in various attack scenarios including single and multiple paths attacks, and attacks from random sources. Our results show that BlockJack is able to handle multiple attacks caused by Autonomous Systems (AS) path changes during a BGP prefix hijacking. In experiment settings with 50 random routers, BlockJack takes on average 0.08 seconds (with standard deviation of 0.04 seconds) to block BGP prefix hijacking attacks.

1 INTRODUCTION

Border Gateway Protocol (BGP)–also known as the Inter-domain routing protocol–is a path-vector protocol that regulates the connectivity and information exchange among Autonomous Systems (AS).[1] Based on data presented in APNIC Research and Development (AS 65000), there are currently almost 70,000 unique ASN seen in their BGP routing table (BGP-Potaroo, 2020). Each AS maintains a number of IP Prefixes (in short Prefixes) and domains assigned by the Internet Assigned Number Authority through Regional Internet Registries (RIR).

To hijack an IP prefix, an adversary router (or AS) advertises a (*fake*) IP prefix that belongs to another router (or AS). When the adversary AS conducts a prefix advertisement, BGP sends the prefix to all neighbours on the Internet. As a result, the traffic that is supposed to reach the original AS, is redirected to the adversary AS which may result in unavailability of Internet services for the targeted AS and other breaches of the security of the Internet traffic. Besides adversarial considerations, instances of IP Prefix hijacking can be due unintentional network mis-configurations (e.g. during network setup (Hope, 2020)).

To prevent BGP hijacking and to enable the validation of the origin of BGP announcements, IETF (the Internet Engineering Task Force) supports the Resource Public Key Infrastructure (RPKI) through RFC 6480. IETF further released RFC 6482 for Route Origin Authorization (ROA) and RFC 6483 for Route Origin Verification (ROV). ROA is a process where an AS authorizes a number of prefixes to be advertised under its jurisdiction, and stores it as a tuple of IP prefix, AS (owner), the maximum length of AS and expiry date of each IP block (Iamartino et al., 2015). To prevent prefix hijacking, the tuple can be utilized during ROV process to verify whether or not an AS advertises the authorized prefix.

The RPKI architecture gradually delegates the process of securing BGP from Internet Assigned Numbers Authority (IANA) as the global regulator to Regional Internet Registrar (RIR), Internet Service Providers (ISPs) and private network companies (Liu et al., 2016). Each of these institutions is allowed to

[a] https://orcid.org/0000-0003-3559-5123
[b] https://orcid.org/0000-0003-0336-0602
[c] https://orcid.org/0000-0003-2714-0276
[1] An AS is an independent network that comprises the Internet and each AS assigned a 16-bit or 32-bit unique number known as Autonomous System Number (ASN).

Sentana, I., Ikram, M. and Kaafar, M.
BlockJack: Towards Improved Prevention of IP Prefix Hijacking Attacks in Inter-domain Routing via Blockchain.
DOI: 10.5220/0010521106740679
In *Proceedings of the 18th International Conference on Security and Cryptography (SECRYPT 2021)*, pages 674-679
ISBN: 978-989-758-524-1

publish a *Certificate of Authority* (CA) to its downstream network and keep the *Resource of Certificate* (RC) in its storage. This process is considered to be rather risky as any attack or a potential miss-configuration on the upstream network can cause the failure of the entire downstream network. This hierarchical structure also provides the privilege for the upstream network to overwrite the RC of its downstream network (Cooper et al., 2013).

We propose `BlockJack`, a consortium blockchain-based system to prevent IP prefix hijacking in Border Gateway Protocol. Instead of providing a centralized repository as in RPKI, BlockJack introduces a distributed repository of IP prefix-ASN pairs in consortium blockchain nodes for prefix authorization and prefix verification purposes. The BlockJack system also stores the consortium member's CA in each AS repository to eliminate the drawback of a centralized or hierarchical public key infrastructure-based scheme.

BlockJack independently runs Blockchain along with BGP network. It creates an interoperable module to synchronise the operations of the Blockchain and BGP network thus, unlike (Liu et al., 2020), BlockJack eliminates the need for router software update. The architecture of BlockJack provides reliability by avoiding the race conditions between the BGP *Update Message* [2] and the Blockchain process, such as prefix authorization or prefix verification. A race condition occurs when the Blockchain transaction runs inside the BGP protocol loop and exceeds the BGP Update Message interval.

To improve robustness of BlockJack's to the dynamic nature of *BGP AS Path* [3] changes and *AS path divergence*–oscillating AS Path changes during the prefix hijacking, BlockJack stores only the prefix and AS origin in the Blockchain. BlockJack uses *Next Hop* [4] information to identify the neighbouring AS that contributes to the addition of prefix in the routing table, in order to create the *Inbound filter*. In case of a hijacking event, an inbound filter blocks the incoming Prefixes from a certain AS through the immediate neighbors. This approach reduces the number of verification and authorization process to the Blockchain caused by dynamic change of AS Path.

The rest of this paper is organized as follows. Section 2 describes the architecture and mechanisms that support BlockJack operations. Section 3 describes Test-bed, BlockJack's test scenarios and analysis of our experiments. While Section 4 presents related work and Section 5 concludes our research.

2 SYSTEM ARCHITECTURE AND MECHANISM

In this section, we present the architecture, process and mechanism of BlockJack.

2.1 System Architecture

BlockJack consists of the following three main modules as depicted in Figure 1:

- **Blockchain.** To regulate the interaction among ASes demanding a highly trusted environment, we leverage *Hyperledger Fabric* (Linux-Foundation, 2020) to build the Blockchain module. Unlike public-based (*permissionless*) Blockchain (such as *Bitcoin or Ethereum*), Hyperledger Fabric eliminates the role of miners in tethering the new blocks to the existing blocks by adding several components.[5] The consensus mechanism ensures only trusted and known consortium members to participate in Blockchain transactions.
- **Profiler.** This module consists of the process for generating the private key, public key, and X.509 certificate for admin and routers in each AS. To store these credentials, Profiler offers a wallet and provides a REST based API to share the credentials among different (sub)modules of BlockJack.
- **Dispatcher.** This module conducts routine tasks to monitor the routing tables and dispatches filtering commands if there is any updates about the BGP routing tables.

2.2 BlockJack's Mechanisms

BlockJack consists of supporting (control) and main mechanisms. Supporting mechanism operates the Blockchain network, creates credentials, initializes the Blockchain, and registers an admin router for control and monitoring purpose. In the following, we

[2] *BGP Update Message* is a signal used to refresh the content of BGP routing tables. The default Update message interval for Cisco router is 30 seconds (Vinit and Brad, 2018).

[3] AS Path is a parameter that stores the list of AS and use to retrieve the prefix source.

[4] *Next Hop* is a parameter used to identify the immediate router in a given AS Path.

[5] Hyperledger Fabric provide *Orderer, Commiter, Chaincode, Endorser and Fabric CA* to support the blockchain transaction mechanism known as *consensus* (Linux-Foundation, 2020).

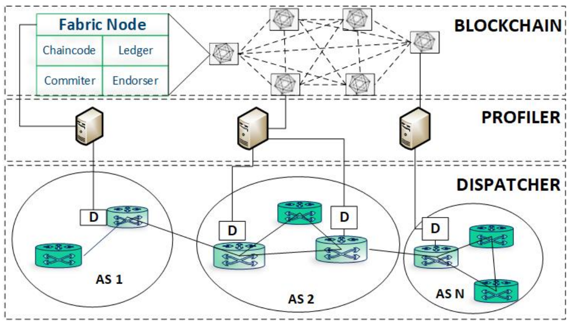


Figure 1: An overview of the three modules of BlockJack. The Blockchain module is handling data storage and data query while Profiler acts as a bridge between Blockchain and Dispatcher as well as to store the Credentials of each router. Dispatcher *monitors* routers and *dispatches* filter commands if Prefix Hijacking occurs.

provide more details about the *main* mechanisms in BlockJack:

- **Prefix Authorization.** This mechanism validates and proves that the prefix belongs to specific ASes. To claim the ownership of a Prefix, an AS first starts the Dispatcher module to read the BGP routing table and captures the *valid-best* route from the prefixes list. If a new prefix is found, Dispatcher sends an HTTPS request to the Profiler asking for prefix addition to the Blockchain. In the Profiler, the REST API server receives the request and authenticates the request to the router profile. The Profiler then adds the router credentials to the request and sends it to the Blockchain. In the Blockchain, Endorser verifies the request based on the contract written in the Chaincode. Once the request complies with the smart contract, the Endorser sends the prefix to the Orderer and then initiates the consensus mechanism in the Blockchain. The Orderer module distributes the requests to all consortium members and gets the approval from at least 50% of the consortium members. When consensus is reached, the Orderer signals the Committer to seal the transaction and adds a new block into the Blockchain ledger. At the end of the Blockchain transaction, the Orderer creates a new *World State* [6] and distributes it to the whole blockchain nodes.

- **Prefix Verification.** Prefix Verification module aims to check whether the announcement received by the router from its neighbors contains the authorized prefix. For this purpose, `BlockJack` compares the prefixes found in the routing table with the prefixes stored in the Blockchain. Prefix Verification mechanism starts when the Dispatcher Module finds a new prefixes announcement from the router's neighbor, and then requests for verification to the Blockchain through Profiler. In the Profiler, Rest API server conducts authentication against the router credentials, and then sends the request to the Blockchain. In the Blockchain, the Chaincode verifies the compliance of the request and then query the Ledger. The query result publishes three kinds of signals; *Valid, Invalid,* and *Unknown*. The Valid signal indicates that the prefix sent to the Blockchain is available and corresponds to the legitimate AS Number. While, an Invalid signal denotes that the prefix exists in the Ledger but corresponds to different AS Number. This invalid signal can indicate that a BGP prefix hijacking occurs in the network. The Unknown signal indicates neither the prefix nor the AS Number are available in the Ledger. When the Dispatcher accepts the Invalid signal, then it produces Inbound filter commands to block the announcement from the AS that was indicated as a source of the prefix hijacking.

[6]*World State* is a current state of database (Ledger) in blockchain.

3 TESTBED SETUP AND EVALUATION

This section presents our experimental testbed and analysis.

3.1 Experiment Testbed

Blockchain module of BlockJack is implemented using Hyperledger Fabric. Each Blockchain node, including *Orderer, Fabric CA,* and *Chaincode* are running in a separated Docker container. For the Profiler module, we tailor *Node JS* to create the admin function, router registration, and REST API server. Using *Python* and *shell script*, we develop the Dispatcher module. While develop BGP network environment leveraging *Quagga* router software, and for this research, we use *Dockerized Quagga image* produced by (Chiodi, 2020). In our testbed, each router runs in a separate *detach-mode* Docker container connected by a customized virtual network. We use the testbed and conduct several sets of experiments to evaluate the performance and resiliency of BlockJack against BGP hijacking attacks. We provide further details in the following:

Performance Evaluation. We aim to evaluate the performance of our BlockJack in terms processing time required for handling prefix *authorization* and *verification* requests. To this end, we conduct two sets of experiments [7] and record the processing time of the authorization and verification mechanisms of BlockJack. In particular, we generate sets of random prefixes from a BlockJack router and query the Blockchain ledger to determine the authorization and verification times required by our proposed systems.

For the prefix authorization setup, we create a function in the Dispatcher module to send a various number of prefixes into the Blockchain to measure BlockJack performance in handling prefix authorization. Each prefix authorization process followed by *commit* order so each prefix addition will be adding a block in the Blockchain. At the end of the experiment there will be a thousand blocks in the chain. To measure the prefix verification time, we create a script so the Dispatcher can send a various number of prefixes to be verified by Blockchain with 1000 blocks.

Resiliency Evaluation. Resiliency evaluation aims at observing the BlockJack's neutralizing capabilities against the prefix hijacking attacks. To measure the resiliency of BlockJack, we leverage Quagga and Docker in the higher computing environment[8] and create several network topologies with various number of routers. We measure the prefix hijacking neutralization in two stages including Prefix Prepending and Neutralization. Prefix Prepending is the process of adding an ASN to the AS-path parameter in the BGP table for each AS passed by a prefix. In this experiment, the prepending time is equal to the time needed by the adversarial prefix to arrive at the router where the Dispatcher resides (BlockJack router) and disrupts the original prefix as the route with the *valid-best* status. While, Neutralization is the stage where BlockJack detects, verifies, and sends filter commands to the router to neutralize hijacking. By measuring the prepending (BGP hijacking attack) time and neutralization (blocking) time, we determine the duration of BGP hijacking attacks and the efficiency of BlockJack to neutralize the attacks, respectively.

For this experiment, we create the following three different scenarios as:

- **Single Path Attack Scenario.** We create a binary tree-like network topology and deploy the dispatcher on the router which is located at the root. We prepared five adversarial prefixes that would be used by routers located in the farthest branch to hijack the prefixes announced by routers in the leaf of the tree. These attacks create single paths when they reach the BlockJack router i.e., root of the tree.
- **Multiple Path Attack Scenario.** This scenario is designed to examine BlockJack's resilience in anticipating routing path changes that occur during BGP prefix hijacking. In this scenario, we modify the binary tree network topology in the first scenario by setting up BGP peering for each branch at the same level. This will cause each announced prefix to have more than one path when it reach BlockJack router at the root of the tree.
- **Random Attack Scenario.** This scenario is set up to examine the BlockJack resilience in a very random BGP environment. We made several random network topologies with various numbers of routers. The connectivity level in each experiment was set to 25% indicating that a node has a probability to be connected to 25% of the total nodes on the network. For each set of experiments, for realistic evaluation, we randomly place the Dispatcher and five random adversarial prefixes in our testbed.

Limitations. We collect 25 experimental results for each scenario in networks consisting of 20 to 60 routers. We observe that our testbed suffers with the increase in number of routers. Docker network shares the same Linux kernel to handle all virtual networks running on top of the Docker container. This condition causes an overload on the virtual network. It also causes the Hyperledger failure to install the Chaincode (smart contract) on the Blockchain node through the corresponding port.

Table 1: Time recorded by BlockJack to *authorize* and *verify* prefixes. Each prefix addition followed by commit process and create a new block.

# of Pref.	Authorization		Verification	
(Block)	Avg. (s)	Total (s)	Avg. (s)	Total (s)
100	2.16	216.21	0.1	10.27
500	2.15	1,076.61	0.09	47.38
1,000	2.15	2,154.32	0.09	92.12

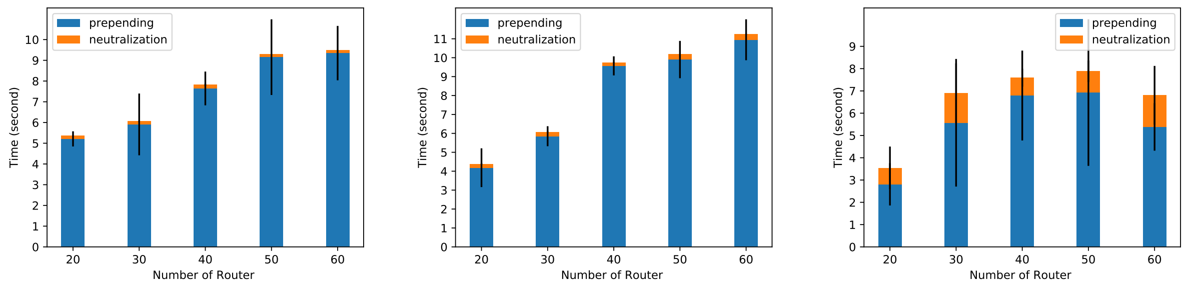
3.2 Analysis

Performance Analysis. This Section presents, the performance analysis of BlockJack in terms of time required for prefix authorization and verification:

Prefix Authorization Analysis. Table 1 shows prefix authorization time increase gradually according to the number of prefixes sent to the Blockchain. On average, BlockJack needs 2,154.32 seconds to authorize 1,000 prefixes with an average authorization time of 2.16 seconds. This heavy or expensive process is caused by a complex consensus mechanism during

[7]Experiments were performed on a workstation with Ubuntu 18.04, CPU 2.7GHZ, and RAM 16GB.

[8]Experiment conduct in Cluster Server with 4 core CPU, 128 GB of memory, 500 GB HD running on Ubuntu 18.04.



(a) Single Path Attack. (b) Multiple Path Attack. (c) Random attack.

Figure 2: Resiliency evaluation results. The prepending time record increases gradually as the number of routers increases in single path and multiple path attack scenarios. While, the prepending time for random attacks fluctuates. All prepending times are presented in the form of time / 10 seconds.

the block addition to the chain.

Based on that result, if the BGP update interval is 30 seconds, the maximum prefixes that can be authorized in one interval is 13 prefixes which is higher than 12 prefixes average announcement per AS origin as shown in (Tony Bates, 2020). Hence, in this case, BlockJack can handle the average prefixes announcement in real world condition in one BGP update message interval simultaneously without the assistant of the local ROA cache, disregarding the network traffic delay.

We then compare this prefix authorization result to the real condition of the prefix ownership. According to (Tony Bates, 2020), the highest number of prefixes announced by an AS is recorded by AS8151 (Uninet S.A. de C.V., MX) with 8125 prefixes. In this case, BlockJack needs 625 BGP update interval or 17,550 seconds to authorize the whole prefixes when it runs for the first time which potentially creates a race condition between BGP Update interval and prefix authorization if the Blockchain node embeds inside the router machine and depends on the BGP message signal. The time taken to access Blockchain during the authorization process far exceeds the BGP update message interval. Hence, the Blocjack's decoupling of Blockchain and routing environment helps in resolving any potential race-conditions.

Prefix Verification Analysis. The experiment result in Table 1 shows that the prefix verification process is much lighter than that of prefix authorization. For instance, BlockJack needs 92.12 seconds to verify 1,000 prefixes or on average of 0.09 second per prefix. Given the worst case scenario that the growth projection of 150 prefixes per day (Geoff, 2020) appear in the concurrent time, and assume that the same amount of prefix withdrew in the same time, then BlockJack only need 27 seconds to verify the 300 updated prefixes. This record is below the 30 seconds BGP Update message interval. Hence, in this case BlockJack can handle Prefix verification without local ROV cache assistance.

However, verifying the global BGP routing table would be so challenging to conduct in one BGP update message interval. Expect that the number of the global routing table is 850,000 (BGP-Potaroo, 2020) and assume that the verification time per prefix is 0.09 seconds, then the total time needed by BlockJack to verify those prefixes on the first time running is 76,500 seconds. That result is equal to 2,550 of the BGP UPDATE message interval. Hence, the existence of a local ROV cache is crucial to reduce the verification request during the BlockJack operation. The Dispatcher can compare the entry of BGP routing table and the local ROV cache to find a new prefix announcement or withdrawal occurrences, and then verify those prefixes. As an addition, any verification that exceeds the BGP message interval is not impacting the BlockJack because the Dispatcher is not dependent on any BGP message signal.

Resiliency Analysis. BlockJack is able to neutralize all adversarial prefixes that disrupt the BGP routing table in the three attacking scenarios. The results of BlockJack's resilience evaluation are depicted in Figure 2. We observe that the average prepending time increases gradually in accordance with the router addition in single path and multiple path attack scenarios, while the average prepending time for random attacks seems to fluctuate. The lowest prepending times for each single path, multiple path and random attack scenario were 28.068, 41.855, and 52.101 seconds, which are recorded during the experiment with 20 routers. The average prepending time for all experimental sets on single path, multiple paths and random attack scenarios, respectively, is 74.527, 80.9088, and 54.9572 seconds.

Neutralization time looks constant in all scenarios with an average of 0.1516 seconds in single path scenarios and 0.2362 seconds in multiple path scenarios, except for random attack scenarios which record an average neutralization time of 1.0484 seconds. The amount of neutralization time in random attack scenarios is up to 5 times compared to multiple path scenarios. This is because the number of neighbors in the random attack scenario is greater than multiple path scenarios. From the five adversarial prefixes sent, the average number of attacks on the random scenario reaches 10.08 attacks, compared to 4.52 attacks received by BlockJack routers in multiple path attack scenarios. If we take a sample of random attack scenario with 50 routers, on average BlockJack needs 0.957 seconds to neutralize 12.04 attacks of five experiment attempts. That record is equal to 0.08 seconds to neutralize a single attack.

The standard deviation of prepending and neutralization time is also seen constant in single path and multiple path scenarios with an average range of 8.2488 seconds to 11.6154 seconds for prepending time and 0.0162 seconds to 0.0188 seconds for neutralization time. While the average standard deviation of prepending and neutralization time in the random attack scenario was recorded at 20.3712 seconds and 0.8998 seconds, respectively.

4 RELATED WORK

The closest work to our research is the work proposed by Sfirakis et al., (Sfirakis and Kotronis, 2019). In the paper Sfirakis et al., introduced the concept of a Blockchain-based prefix hijacking prevention using Bitcoin. The proposed system has been tested in Quagga router software. However, this system required the AS owner to provide share coins for every prefix authorization or prefix verification requests, as the blockchain need miners to attach new blocks to the blockchain.

5 CONCLUSION

Although the Prefix Authorization and Prefix Verification processes can ideally be handled in one BGP Update message, several conditions will cause a race condition between processes that occur on the Blockchain and processes that occur in BGP. In this paper we presented how Blockjack addresses this specific issue. BlockJack also manages dynamic-multiple hijacking scenarios due to changes in the BGP attribute values which cause dynamic changes in determining the best-valid path in the BGP routing table.

ACKNOWLEDGEMENTS

The first Author is a Scholarship Awardee of the Indonesian Endowment Fund for Education (LPDP) of the Republic of Indonesia, with the LPDP ID 20193221014024. This research is partially funded by the Optus Macquarie University Cyber Security Hub.

REFERENCES

BGP-Potaroo (2020). Bgp analysis report-bgp table. https://bgp.potaroo.net/index-bgp.html. BGP Table Data last accessed 1 July 2020.

Chiodi, P. C. (2020). Quagga router software code @github. https://github.com/pierky/dockerfiles/tree/master/quagga. last accessed 1 July 2020.

Cooper, D., Heilman, E., Brogle, K., Reyzin, L., and Goldberg, S. (2013). On the risk of misbehaving rpki authorities. In *Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks*, HotNets-XII, New York, NY, USA. Association for Computing Machinery.

Geoff, H. (2020). Bgp in 2019 – the bgp table. https://blog.apnic.net/2020/01/14/bgp-in-2019-the-bgp-table/. Blog APNIC last accessed 1 Juli 2020.

Hope, A. (2020). Russian rostelecom compromises internet traffic through bgp hijacking. https://www.cpomagazine.com/cyber-security/russian-rostelecom-compromises-internet-traffic-through-bgp-hijacking. Accessed: 18/12/2020.

Iamartino, D., Pelsser, C., and Bush, R. (2015). Measuring bgp route origin registration and validation. In Mirkovic, J. and Liu, Y., editors, *Passive and Active Measurement*, pages 28–40, Cham. Springer International Publishing.

Linux-Foundation (2020). Hyperledger fabric release 2.0. https://hyperledger-fabric.readthedocs.io/en/release-2.0/whatis.html. Hyperledger Fabric Release 2.0 readthedocs last accessed 1 July 2020.

Liu, X., Yan, Z., Geng, G., Lee, X., Tseng, S.-S., and Ku, C.-H. (2016). Rpki deployment: Risks and alternative solutions. In Zin, T. T., Lin, J. C.-W., Pan, J.-S., Tin, P., and Yokota, M., editors, *Genetic and Evolutionary Computing*, pages 299–310, Cham. Springer International Publishing.

Liu, Y., Zhang, S., Zhu, H., Wan, P.-J., Gao, L., Zhang, Y., and Tian, Z. (2020). A novel routing verification approach based on blockchain for inter-domain routing in smart metropolitan area networks. *Journal of Parallel and Distributed Computing*, 142:77 – 89.

Sfirakis, I. and Kotronis, V. (2019). Validating IP prefixes and as-paths with blockchains. *CoRR*, abs/1906.03172.

Tony Bates, Philip Smith, G. H. (2020). Cidr report for 1 jul 20. https://www.cidr-report.org/as2.0/. CIDR Data report last accessed 1 July 2020.

Vinit, J. and Brad, E. (2018). *BGP Message*. Cisco Press, San Fransisco.