UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

Securing the Internet at the Exchange Points

Tomás Joaquim Gonçalves Peixinho do Vale

Mestrado em Engenharia Informática
Especialização em Arquitectura, Sistemas e Redes de Computadores

Dissertação orientada por:
Prof. Doutor Alysson Neves Bessani
Prof. Doutor Fernando Manuel Valente Ramos

2022

Acknowledgements

First and foremost, I would like to thank both my thesis advisors, Professor Fernando Ramos and Professor Alysson Bessani. Finishing this thesis has been a long and arduous process, having taken more time than probably any other master's thesis ever. That being said, both of them always supported me and allowed me the time and space I required in order to complete it, giving me advice and technical support whenever I needed it. A big thank you to them both for the patience and support throughout all this time.

I would also like to thank my friends and family for believing in me, and supporting me all this time, with a special thanks to three people in particular. The first is my mother, who truly believed in me and pushed me to not give up with all of her heart. If not for her constant support and motivation, I might have given up several times. The second is my good friend Afonso Mendes, who guided me through some tough times, gave me advice regarding the writing of the thesis, and even helped me with some of the graphs. And last but certainly not least, my good friend Francisco Farinha. His help was imperative. Without his help and much-needed pushing I wouldn't have been able to finish the actual project of this thesis. I owe it all to him.

Thank you all once again, I couldn't have done it without you!

Resumo

O BGP, ou *Border Gateway Protocol*, é o principal protocolo de roteamento entre Sistemas Autónomos da Internet. Apesar de ser vastamente utilizado, tem vários problemas de segurança intrínsecos, e é frequentemente alvo de ataques. Estes ataques são conhecidos como *Prefix Hijacks* (roubo de prefixos) e *Path Manipulations* (manipulação de rotas). Um roubo de prefixo ocorre quando um sistema autónomo anuncia um prefixo IP que não é o seu. Uma manipulação de rota ocorre quando um sistema autónomo anuncia, para um determinado prefixo IP, um caminho que não foi anunciado e que, portanto, não é legítimo. Estes ataques perturbam o correcto funcionamento da Internet através do redireccionamento de tráfego, permitindo que este seja espiado ou manipulado por um atacante, ou seja impedido de chegar ao seu destino, afectando a disponibilidade.

Estes problemas resultam da falta de mecanismos de segurança inerentes ao BGP, que validem a informação de roteamento anunciada pelas diversos sistemas autónomos que formam a Internet. Mais especificamente, este protocolo não autentica a origem de um prefixo, nem valida as rotas anunciadas. Isto significa que uma rede que intercepte um anúncio BGP pode anunciar de forma maliciosa um prefixo IP que não possua, ou inserir um caminho falso para um prefixo, com o objectivo de interceptar o tráfego. Em casos extremos, estes ataques podem levar a uma situação de *Blackholing*, em que o tráfego, ao ser redireccionado por caminhos fictícios, é impedido de chegar ao prefixo de destino.

Várias soluções foram propostas no passado, no entanto todas com inúmeras limitações, das quais a mais severa sendo a necessidade de alterações drásticas à infraestrutura actual do BGP (isto é, a necessidade de troca do equipamento existente). Para além disto, a maioria das soluções requerem a implementação generalizada por parte dos múltiplos sistemas autónomos da internet, de forma a tornarem-se eficazes. Finalmente, estas soluções necessitam tipicamente de canais de comunicação seguros entre os routers participantes, o que requer capacidades de verificação criptográficas computacionalmente intensivas e que normalmente não estão disponíveis neste tipo de equipamentos. Actualmente, as soluções mais adoptadas têm estes requisitos em conta, no entanto acabam por ter de chegar a um compromisso entre as garantias de segurança que conseguem oferecer e a dificuldade da sua adopção.

Com estes desafios em mente, esta tese propõe-se a investigar a possibilidade de melhorar a segurança do BGP através da utilização de tecnologia de rede definida por soft-

ware (isto é, a chamada *software-defined networking*, ou simplesmente SDN), que tem sido cada vez mais utilizada nos *Internet Exchange Points* (IXPs). Estas instalações de interconexão são localizações únicas que tipicamente ligam centenas ou até mesmo milhares de redes, funcionando como "intermediários" de Internet idealmente posicionados para implementar mecanismos inter-rede, tais como segurança, sem necessitarem de alterações à infraestrutura dos operadores de rede.

A nossa ideia chave trata de incluir um canal seguro entre IXPs que, ao correr no servidor SDN que controla estas infraestruturas modernas, evita os requisitos criptográficos nos routers. Na nossa solução, o canal de comunicação seguro implementa uma *ledger* distribuída (a chamada tecnologia de *Blockchain*), de forma a fornecer confiabilidade descentralizada, entre as outras inúmeras garantias que esta tecnologia oferece. Para além disto, a tecnologia de *Blockchain* oferece ainda a possibilidade de segurança programável, através da implementação de *Smart Contracts* (ou Contratos Inteligentes), isto é, contratos programáveis que todos os participantes da rede são obrigados a cumprir. A lógica por detrás desta decisão prende-se com o facto de que, ao aumentar a confiabilidade e evitando *updates* caros e extensivos à infraestrutura corrente, pretendemos criar incentivos para os operadores de rede aderirem a estes novos serviços de segurança baseada em IXPs.

Neste sentido, a solução proposta foi implementada utilizando o *Hyperledger Fabric* como *framework* de desenvolvimento de *blockchain*. O algoritmo da solução foi testado utilizando *datasets* com anúncios BGP reais retirados de *route collectors*, que posteriormente foram populados com anúncios maliciosos aleatoriamente inseridos. O rácio entre entradas maliciosas e não maliciosas que foram detectadas pelo algoritmo permitiu-nos apurar a eficácia da solução desenhada. Os testes de performance foram executados utilizando uma rede simulada, com os participantes necessários ao nível da internet, desde IXPs a ASes, bem como todos os componentes de uma *Blockchain*, nomeadamente os diferentes tipos de *peers*, as autoridades de certificação, os canais, etc. A testagem nestas duas vertentes permitiu-nos investigar a viabilidade duma solução deste tipo, bem como comparar a possibilidade da sua implementação no mundo real com outras soluções já existentes.

Os resultados principais deste estudo permitiram-nos compreender melhor o aperfeiçoamento de segurança que seria esperado com esta abordagem radical, mas realista, bem como perceber alguns dos desafios de escalabilidade que esta implementação levantaria.

Palavras-chave: BGP, IXP, SDN, AS, segurança, protocolo de roteamento, roubo de prefixo, manipulação de rotas, blockchain, smart contract, Hyperledger Fabric

Abstract

BGP, the border gateway protocol, is the inter-domain routing protocol that glues the Internet. Despite its importance, it has well-known security problems. Frequently, the BGP infrastructure is the target of prefix hijacking and path manipulation attacks. These attacks disrupt the normal functioning of the Internet by either redirecting the traffic, potentially allowing eavesdropping, or even preventing it from reaching its destination altogether, affecting availability.

These problems result from the lack of a fundamental security mechanism: the ability to validate the information in routing announcements. Specifically, it does not authenticate the prefix origin nor the validity of the announced routes. This means that an intermediate network that intercepts a BGP announcement can maliciously announce an IP prefix that it does not own as theirs, or insert a bogus path to a prefix with the goal to intercept traffic.

Several solutions have been proposed in the past, but they all have limitations, of which the most severe is arguably the requirement to perform drastic changes on the existing BGP infrastructure (i.e., requiring the replacement of existing equipment). In addition, most solutions require their widespread adoption to be effective. Finally, they typically require secure communication channels between the participant routers, which entails computationally-intensive cryptographic verification capabilities that are normally unavailable in this type of equipment.

With these challenges in mind, this thesis proposes to investigate the possibility to improve BGP security by leveraging the software-defined networking (SDN) technology that is increasingly common at Internet Exchange Points (IXPs). These interconnection facilities are single locations that typically connect hundreds to thousands of networks, working as Internet "middlemen" ideally placed to implement inter-network mechanisms, such as security, without requiring changes to the network operators' infrastructure. Our key idea is to include a secure channel between IXPs that, by running in the SDN server that controls these modern infrastructures, avoids the cryptographic requirements in the routers. In our solution, the secure channel for communication implements a distributed ledger (a blockchain), for decentralized trust and its other inherent guarantees. The rationale is that by increasing trust and avoiding expensive infrastructure updates, we hope to create incentives for operators to adhere to these new IXP-enhanced security services.

The main results of this study allowed us to better understand the security improvements that would be attained with this radical yet realistic approach, as well as to understand some of the scalability challenges of its implementation.

Keywords: BGP, IXP, SDN, security, routing protocol, prefix hijack, path manipulation, blockchain

Contents

List of Figures

List of Tables

List of acronyms

AS	Autonomous System
ASN	Autonomous System Number
BGP	Border Gateway Protocol
BGPSec	Border Gateway Protocol Security
BH	Blackholing
BC	Blockchain
CA	Certificate Authority
CC	Chaincode
CLI	Command Line Interface
CSCC	Configuration System Chaincode
DDoS	Distributed Denial of Service
DoS	Denial of Service
DNS	Domain Name System
ESCC	Endorsement System Chaincode
FN	False Negatives
FP	False Positives

HLF	Hyperledger Fabric
IP	Internet Protocol
ISP	Internet Service Provider
IXP	Internet Exchange Point
LSCC	Lifecycle System Chaincode
MOAS	Multiple Origin Autonomous System
MSP	Membership Services Provider
MVCC	Multiversion Concurrency Control
NLRI	Network Layer Reachability Information
OA	Origin Authentication
PH	Prefix Hijack
PKI	Public Key Infrastructure
PM	Path Manipulation
PV	Path Validation
QSCC	Query System Chaincode
RC	Route Collector
RIR	Regional Internet Registries
RIS	Routing Information Service
ROA	Route Origin Authorization
RPKI	Resource Public Key Infrastructure

RS	Route Servers
S-BGP	Secure Border Gateway Protocol
SC	Smart Contract
SDN	Software-Defined Network
SDX	Software-Defined Exchange Point
soBGP	Secure Origin Border Gateway Protocol
SSL/TLS	Secure Sockets Layer/Transport Layer Security
TN	True Negatives
TP	True Positives
VSCC	Validation System Chaincode

Chapter 1

Introduction

BGP is the inter-domain routing protocol of the internet that allows networks (Autonomous Systems or ASes) to announce reachability information to their peers. Inter-domain peering is established between ASes directly or through Internet Exchange Points (IXPs). In this thesis we investigate the potential of a solution to improve the security of BGP that recognizes the increasingly central role IXPs play in the Internet, and as such proposes to leverage *IXP collaboration* to improve Internet security. This chapter provides an introduction to the thesis, including the lack of BGP security that motivates it, as well as the objectives of our work. It also presents the main contributions and the organization of the document.

1.1 Motivation

The main motivation of this thesis are the decades-old BGP security issues, which have led to several attacks over the years, including prefix hijacks and path manipulation. To understand the problem, let us give some context. Autonomous Systems (ASes) periodically send BGP Updates to their neighbors. Since these updates have no integrity protection, a malicious actor can intercept an announcement and easily create a bogus update, enabling those two types of attacks.

These attacks can lead to two main consequences: traffic attraction or traffic interception. Traffic attraction occurs when an AS falsely claims to be the origin of a prefix, leading to blackholing, where traffic is discarded and never reaches its final destination. Traffic interception is similar but in this case traffic is only intercepted, possibly for spying or tampering, but still reaches its destination.

One of the most notable solutions to BGP security problems is the resource public key infrastructure [4]. The RPKI ensures origin authentication through the use of cryptographically verified route origin authorizations (ROAs). A ROA certifies a certain AS as the origin of a prefix, guaranteeing that an AS cannot forge a BGP announcement and make the claim it owns that prefix in order to hijack its traffic. The main problems with

the RPKI are, on the one side, that it does not prevent path manipulation. In addition, as is common in many solutions, it requires wide-scale adoption for it to be effective, which is not the case today.

To deal with path manipulation, BGPSEC was proposed [5]. This standard incorporates the RPKI and replaces the AS_PATH attribute in BGP Update messages with a BGPSEC_PATH that can be cryptographically verified. As a result, a malicious actor cannot change the announced path, precluding interception attacks. This, however, also has a few major roadblocks. Specifically, the need to replace the current routers with BGPSEC capable routers and, as with the RPKI, the need for widespread adoption before it can become a useful solution.

For these reasons, mechanisms to secure BGP problems are still not widely deployed to this day. We were therefore motivated to investigate solutions that can be more easily deployed, namely by including incentives for adoption and by avoiding costly infrastructure upgrades.

1.2 Objectives and Solution

The purpose of this thesis is to investigate inter-IXP collaboration as a possible solution for enhanced BGP security, with a focus on ease of deployment and creation of incentives. IXPs are well placed for such solution due to their central role in interconnecting many networks. By centering the solution in the IXPs, peering networks are not required to change their infrastructure to implement the new security mechanisms.

Our security goal is to mitigate prefix hijacks and path manipulations. The key idea is to implement origin authentication and path validation mechanisms at the inter-IXP level. The design leverages the centralized, software-defined nature of modern IXPs, also known as Software Defined Exchanges (SDX) [6, 7]. The solution includes a secure communication channel between these entities. The use of SDN technology enables avoiding changes to the current internet infrastructure: existing routers and switches perform traffic forwarding and routing, while the SDN controllers intercept the required BGP control traffic to run the traffic verification logic.

A blockchain is the proposed solution for implementing the secure communication channel, for various reasons. First, it provides cryptographic capabilities and decentralized trust, through its use of distributed ledger technology. Other guarantees include immutability and non-repudiation of updates, adding more trust to hopefully foster adoption. Path validation and origin authentication can be ensured through the implementation of such mechanisms in the form of smart contracts, the programmable logic of a blockchain. Smart contracts define the rules and penalties around an agreement in the same way that a traditional contract does, while also automatically enforcing those obligations, thus guaranteeing that no participant is able to avoid them. This enhanced visibility and assured

trust can create incentives that may potentially lead to a wider adoption by the peering community.

1.3 Contributions

This thesis provides three main contributions:

- It proposes an architecture that employs SDN and blockchain technology at the IXP level, to enable secure inter-IXP communication for improved BGP security;
- It proposes an algorithm and implements it in the form of a smart contract, to offer origin authentication and path validation of BGP updates at the inter-IXP level;
- It evaluates the performance and effectiveness of the solution by running experiments using a cluster of servers for the former, and by employing datasets based on real BGP traces for the latter.

1.4 Organization

The rest of the document is organized as follows: Chapter 2 provides the background for this thesis; Chapter 3 presents the design and implementation details of the solution; Chapter 4 details the evaluation of the solution and discusses its results; Chapter 5 concludes this thesis.

Chapter 2

Background and related work

This chapter presents related work in the context of this thesis. We start by providing a general overview of BGP and by detailing some of the existing security issues that BGP faces, as well as some of the current solutions to these problems. We also provide an introduction to blockchain technology and some frameworks, namely the Hyperledger Fabric. Finally, we present solutions that have proposed blockchain technology in the Internet context.

2.1 Internet Routing

In this section, we present an overview of BGP, the role IXPs are playing in the current internet routing infrastructure, and their inherent value when it comes to applying software-defined networking technology to improve BGP security.

2.1.1 BGP Overview

The Internet is a system of interconnected networks throughout the world that uses BGP [8] as its main inter-domain routing protocol. It is through this protocol that these networks, or autonomous systems (ASes), announce their presence to their neighbors. These Autonomous Systems typically consist of a set of routers that belong to an ISP (internet service provider).

Each AS has a unique AS number (ASN), which identifies it, and is associated with a set of IP addresses (IP prefixes) that it owns. There are two types of business relationships ASes can have: customer/provider (customer pays the provider to send and receive traffic) and settlement-free peering (two ASes agree to transit each other's traffic for free). Finally, ASes can be Stubs (communication endpoints with connections to the rest of the internet only through a single upstream provider), multihomed (similar to stub ASes but with multiple upstream providers), and Transit (have connections to multiple ASes and allow stub traffic to flow through toward the Internet core).

BGP is a path-vector protocol [9] that includes the path of ASes to use to reach the prefix being announced (AS_PATH as one of the UPDATE message fields). BGP peers constantly exchange this Network Layer Reachability Information (NLRI), including known paths and prefixes, in order to provide all ASes the means to forward their traffic correctly. The BGP-4 protocol (its current version) establishes five types of messages: OPEN, UPDATE, KEEPALIVE, NOTIFICATION, ROUTE REFRESH.

The BGP UPDATE message is the most important for this work and in general, as it includes the announcements that carry the routing information. It is by looking at this type of message that networks are informed of the paths of ASes through which they can forward their traffic to reach the desired IP prefixes.

In the next section, a description of IXPs is provided, as a way to introduce their importance in the design of the solution.

2.1.2 Internet Exchange Points

IXPs, or Internet Exchange Points, serve as a type of internet middlemen, that connect multiple ASes. IXPs are a key part of the Internet ecosystem today, and represent a vital way to increase the affordability and quality of connectivity in local communities. They are usually dispersed across countries to enable local networks to efficiently exchange information. They create efficient interconnection points that encourage network operators to connect in the same location in search of beneficial peering arrangements, cheaper and better traffic exchange, and other information and communication services.

An IXP consists of two major components, namely a switching fabric (data plane) and a route server [10] (the control plane). Route Servers enable peering at scale, that is, with a single BGP session a member can establish peerings (exchange routes and interconnect) with all other Route Server users. IXPs are becoming fundamental infrastructures for Internet Routing. They exist in 116 [11] countries, and the total number of IXPs has already surpassed 700 [11]. The number of IPv4 prefixes (set of IP addresses) routed by these IXPs is now beyond 22.2 million [11].

IXPs, with their strategic positioning and large-scale peering, can be an important vantage point to implement solutions to increase BGP security.

2.1.3 Software-Defined Exchanges

SDN, or software-defined networking, refers to the decoupling of network control and forwarding functions (in other words, the separation of the control and data planes). Moving the control plane to a logically centralized, remote controller, allows for full programmability of that plane. More specifically, network applications such as routing or firewalling run on top of the controller, which greatly simplifies the development and testing of new protocols, and also opens the doors to innovative solutions for BGP security

issues. With SDN, enterprises, data center, and service providers can apply diverse actions on packets based on multiple header fields, and exercise "remote control" from a vantage location over packet handling. This flexibility enables applications such as inbound traffic engineering, redirection of traffic to middleboxes, wide-area server load balancing, blocking of unwanted traffic, among others.

SDN is also permeating IXPs, in the form of Sofware-Defined Exchanges. SDX targets large-scale IXPs to offer solutions for scalability, allowing incremental deployments and interoperation with legacy systems [7]. Besides working as route servers, some SDX controllers today already incorporate security information such as RPKI records [4], to enable routing decisions that are based on a richer set of information than in today's BGP routing.

After summarizing the current Internet infrastructure, in the next section we detail the most important BGP security issues – the core motivation of our thesis.

2.2 BGP Security Issues

IP prefixes are owned, and therefore originated, by a specific Autonomous System. In BGP, each router maintains updated routing tables with information on the neighbors to which they should forward packets to each destination prefix, by exchanging regular updates. The major vulnerabilities of BGP are that a malicious actor can announce a prefix as its own, or manipulate the announced AS paths, as there are no authentication nor verification mechanisms to prevent it.

The most common attacks on BGP are **prefix hijacks** [12], the illegitimate takeover of IP prefixes, and **path manipulation** attacks. These can occur in three different scenarios:

- An AS announces itself as the origin of a prefix that it does not actually own.
- An AS announces itself as the origin of a more specific prefix of a prefix it does not actually owns. As routers perform longest prefix matching to forward traffic, these subprefix hijacks are often more severe.
- An AS announces that it can route traffic to the hijacked prefix through a different route than the existing ones (e.g., a shorter route), whether or not the route actually exists. This type of attack is usually referred to as path manipulation or route diversion.

These attacks divert traffic from its normal path and can lead to two possible scenarios: traffic attraction or traffic interception [13]. In a **traffic attraction** attack, the traffic ends up on an AS that does not have a path to the destination, and the packets are discarded, affecting availability (also known as a "black hole" attack). In **traffic interception**, traffic traverses through a network that it should not have traversed otherwise, which may allow

the attacker to increase revenue from customers by having the traffic flow through them, but also to drop, tamper, or snoop on the packets they receive. The main difference is that in the first case, packets are lost, whereas in the second, packets can still achieve to their destination, making the attack usually harder to detect.

A well-known example of a prefix hijack was when the Pakistani government, on November 11 2008, wishing to block Youtube access on its country, had its service providers announce from their AS a BGP route for Youtube, which managed to attract all of Youtube's traffic to their AS, knocking Youtube offline for two hours [14] (Figure 2.1).

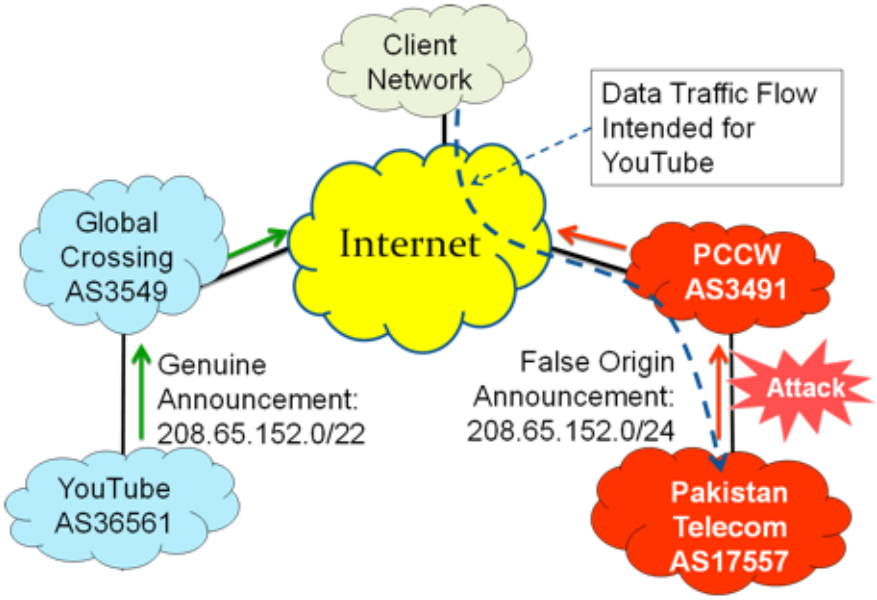


Figure 2.1: Pakistan Telecom announces Youtube's prefix and attracts its traffic [1]

The following section details some of the existing solutions to the aforementioned problems, as well as a brief description of their deployment hurdles.

2.3 BGP Security Solutions

In this section we present architectural solutions to some of BGP security issues. In addition, we present analysis tools and mitigation mechanisms that may increase security without requiring structural changes to the infrastructure.

2.3.1 Architectural solutions

There are four major extensions to increase BGP security. Ordered from weakest to strongest, they are: regular origin authentication [15], soBGP [16], S-BGP [17], and Data Plane Verification [18].

Origin authentication [15] uses a trusted database (e.g., the RPKI) to guarantee that an AS cannot falsely claim to be the rightful owner for an IP prefix. Secure Origin BGP (**soBGP**) [16] also provides origin authentication, to which it adds a trusted database that guarantees that any announced path physically exists in the AS-level topology. Secure BGP (**S-BGP**) [17] adds cryptographically-signed routing announcements to origin authentication, to ensure Path Verification (an AS can only announce a path to its neighbours, if at least one of its neighbours previously announced that same path to the first AS). Finally, **data-plane verification** [18] prevents an AS from announcing one path while forwarding on another.

The most widely deployed security mechanism to BGP is the RPKI (resource public key infrastructure) [4]. This solution allows authentication of an AS when he announces a specific prefix, by pairing ASNs with IP prefixes. As such, it provides origin authentication.

The RPKI has three main building blocks which define what it allows to validate:

- **Trust-Anchors**, which are organizations that handout network resources, also known as RIRs (Regional Internet Registries);
- Cryptographic attestations called Route Origin Authorizations (**ROAs**). ROAs link a set of prefixes with an origin ASN and are signed by the resource holder's private key;
- **Validators**: routers with the ability to validate BGP prefixes against ROAs.

With these three blocks in place, it is possible to verify the origin of an announcement, preventing prefix hijackings.

To tackle path manipulation, **BGPSEC** [5] was introduced as an RFC, as an implementation of a S-BGP mechanism. BGPSEC incorporates the use of the RPKI, thus guaranteeing origin authentication. In addition, BGPSEC replaces the AS_PATH attribute in BGP updates with a new BGPSEC_Path attribute, which performs the same function, but now with integrity guarantees.

These two prominent solutions have, however, two major drawbacks. Both have very low coverage, with the RPKI – the most successful – having only 5% of all prefixes registered. Besides this problem, BGPSEC requires cryptographic verification capabilities at the routers, which are often unavailable. Unfortunately, to be effective these schemes require wide deployment, attesting the importance of *incentive mechanisms* for adoption.

While these architectural changes address the root of the problem, its incipient adoption demands other mechanisms and techniques that, instead of altering the routing protocol or the infrastructure, try to mitigate these problems with ad-hoc mechanisms. These can be separated as monitoring tools and mitigation techniques, as detailed in the sections ahead.

2.3.2 Monitoring Tools

These tools collect data from routers for analysis and troubleshooting. These are reactive approaches since they only act after an attack has already occurred. Two examples of such tools are BGPStream [19] and HEAP (Hijacking Event Analysis Program) [20].

BGPStream [19] is a framework for data analysis of both historical and real-time BGP data. It allows for the investigation of how BGP communities propagate and are visible via the RouteViews and RIPE RIS measurement infrastructures.

These communities can then be used to study several relevant Internet phenomena, such as complex AS relationships, traffic engineering policies, and DDoS mitigation. By studying these behaviours, it enables predicting certain malicious events of the same nature, and potentially prevent them from happenng. This preventive approach, however, requires the existence of historical data with similar attacks that have already happened, so they are ineffective against new attacks.

The other example is **HEAP** [20], a detection mechanism that reasons about elaborate routing attacks given a set of previously defined filters. It tries to identify attacks by providing administrative assurance obtained from IRRs, operational assurance based on common routing practices, and cryptographic assurance from SSL/TLS measurements.

2.3.3 Mitigation Mechanisms

Mitigation mechanisms range from blackholing malicious networks so that no traffic is redirected from/to them, to policy-based filtering at routers. These are proactive approach since they try to prevent attacks before they occur.

Blackholing [21] is used to prevent DDoS attacks at the BGP level. Despite being cost-efficient and effective, it makes the AS under attack unreachable. As such, other solutions have been gaining momentum recently. Examples of such mechanisms include systems like Flowspec [22], ARTEMIS [23], and Stellar [24].

Flowspec [22] is a BGP flow specification feature, which allows the deployment and propagation of fine-tuned filters across AS domain borders. **Stellar** [24] is a more aggressive approach that provides the means to perform fine-grained blackholing at IXPs (limiting the collateral damage of regular blackholing). It achieves this by combining available hardware filters with novel signaling mechanisms, without the need for high-level cooperation to enhance mitigation effectiveness. **ARTEMIS** [23], on the other hand, is a real-time defense approach that is operated directly by the ASes. It uses a local configuration file with information about the prefixes owned by the network and a stream of BGP updates as input. However, it only detects attacks towards prefixes owned by the network running it.

Another recent idea is the profiling of **BGP Serial Hijackers** [25] through analysis of public BGP data, by capturing persistent misbehaviour of specific ASes and defining them

as “bad guys”, and using that information in prevention techniques. By illuminating their routing characteristics and how they differ from legitimate networks, the results can be used in automated applications (using machine learning) or to generate reputation scores.

All these non-architectural solutions are important, practical, and for that reason many are widely deployed. However, they are still “patches”: ad-hoc solutions that address part of the problem, but not its root.

This thesis: Ours is a new architectural approach to improve Internet security. We call it BGPSECx. Our goals are the same as BGPSEC: to provide origin authentication and path validation. To these, we add two additional requirements that we believe are hindering wide adoption of these BGP security mechanisms: *avoid changes to routers* and *create incentives for adoption*. As will be made clear later, we aim to achieve these goals by deploying BGPSECx at IXPs, and by promoting secure inter-IXP collaboration mediated by a blockchain. We give some background on this technology next.

2.4 Blockchain

A blockchain [26] is a shared ledger, which is an open, distributed database that keeps track of who owns an asset (be it financial, physical, or electronic). Participants keep a copy of the blockchain, which is updated every time a transaction occurs, and it is deployed within a peer-to-peer network. It comprises a growing list of records called blocks that contain transactions, which are protected by cryptographic hashes and consensus. The blockchain database is not stored in any single location. Rather, it is decentralized.

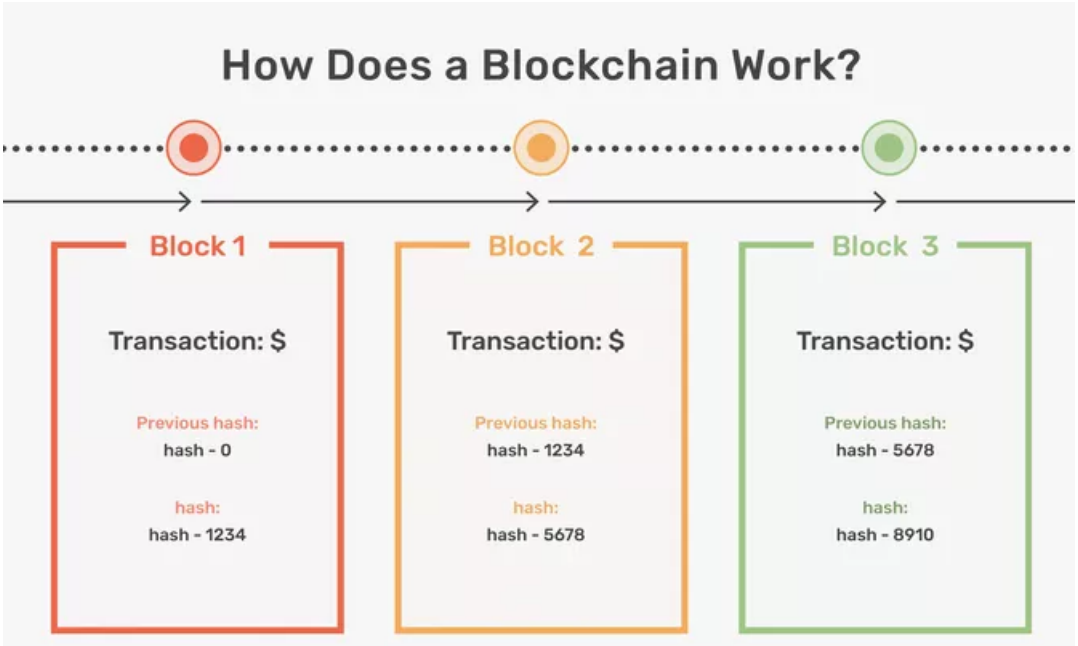


Figure 2.2: Simple Blockchain Example [2]

The original Bitcoin blockchain is permissionless [26], with any node being able to join the network. This blockchain uses (expensive) proof-of-work mechanisms to add transactions to the chain. The other type of blockchains are permissioned [27], where a set of trusted nodes execute byzantine consensus to append new blocks.

Existing blockchain frameworks include both permissioned and permissionless blockchains, for example, Openchain, Corda, Multichain, Ethereum, and Hyperledger Fabric, just to name a few [28].

Ethereum [29] is a public, decentralized, open-source blockchain featuring smart contract functionality, which includes a native cryptocurrency called Ether (ETH). **Openchain** [30], in addition, specifically allows financial establishments to effectively manage all financial processes and enhance anti-fraud and anti-money laundering systems while relying on a partitioned consensus algorithm. Similarly, **Corda** [31] is an open-source technology that can be used to store, manage, control, and synchronize financial obligations between different organizations. **Multichain** [32] is based on the mining principle and offers privacy and openness with a feature for managing user permissions. It is highly configurable and customizable, allowing it to work with several blockchains simultaneously.

Currently, one of the most popular blockchain projects is **Hyperledger** [33], which comprises a range of smart contract engines: Hyperledger Sawtooth, Iroha, Fabric, and Burrow, just to name a few. Its popularity is probably due to its permissioned nature, modular structure, and strong community. We detail the most popular of these engines next.

2.4.1 Hyperledger Fabric in Detail

Hyperledger Fabric [3] is an open-source, enterprise-grade, permissioned, distributed ledger technology platform. This high-performance solution achieves a throughput of more than 3500 transactions per second, and a latency of under one second, on average.

It is permissioned, which means that the participants are known to each other, and it is ensured a certain degree of trust, allowing them to be identifiable, as well as being able to define policies based on said identities. It enables confidentiality through its channel architecture and private data features.

Fabric has a modular and configurable architecture, enabling innovation, versatility, and optimization for a broad range of industry use cases. Its modular building blocks are:

- Consensus protocol/Ordering service [34];
- Identity and membership;
- Scalable dissemination;
- Smart-contract execution;

- Ledger maintenance.

Fabric does not require a native cryptocurrency, which means there's no incentive for any costly mining or to fuel smart contract execution. Since Fabric is a permissioned blockchain, all nodes that participate in the network have an identity, provided by a Membership Service Provider (MSP). Nodes in the network can have one of three roles: clients, orderers, or peers [3].

Clients submit transaction proposals for execution, help orchestrate the execution phase, and broadcast transactions for ordering. **Orderers** are the nodes that collectively run the ordering service, which establishes the total order of all transactions in Fabric. Each transaction contains state updates and dependencies computed during the execution phase, along with cryptographic signatures of the endorsing peers that computed them.

Peers execute transaction proposals and validate transactions, while also maintaining the Blockchain (append-only record of all transactions in the form of a hash chain) and the World State (succinct representation of the latest ledger state). The peers that execute the transaction proposals are called Endorsing Peers (or simply endorsers), as specified by the endorsement policy of the chaincode to which the transaction pertains. Not all peers are endorser peers but they all maintain the complete ledger.

In Fabric, a distributed application consists of a Smart Contract and an Endorsement Policy [3]. A smart contract is the code that defines the application logic, known as Chaincode. It is a program written in any of a wide array of general-purpose programming languages that implements a prescribed interface. Chaincode initializes and manages a ledger state through transactions submitted by applications. A chaincode typically handles business logic agreed to by members of the network.

An endorsement policy specifies the set of peers on a channel that must execute chaincode and endorse the execution results in order for the transaction to be considered valid. These endorsement policies define the organizations (through their peers) who must "endorse" (approve of) the execution of a proposal.

The Fabric chaincode lifecycle [3] requires that organizations agree to the parameters that define a chaincode, such as name, version, and the chaincode endorsement policy. The lifecycle can then be divided into four stages.

First, the chaincode needs to be **packaged**. Then, the chaincode package needs to be installed on every peer that will execute and endorse transactions. The peer will build the chaincode after it is installed. Organizations (structures that encompass multiple peers) should only package a chaincode once and then install that same package on every peer they control.

After that, the chaincode is **created** and **initialized** on a channel. This is a chaincode channel binding process called instantiation. The instantiation also sets up the endorsement policy for that chaincode on that channel. With this, the chaincode enters the active state on the channel and is ready to process any transaction proposals, with the transac-

tions being processed concurrently as they arrive at the endorsing peer.

Finally, a chaincode may be **upgraded** at any time by changing its version. Other parts, such as owners and instantiation policy, are optional. However, the chaincode name must be the same, otherwise, it would be considered as a totally different chaincode.

Fabric provides a blockchain architecture aiming at resiliency, flexibility, scalability, and confidentiality, that follows an execute-order-validate paradigm for distributed execution of untrusted code in an untrusted environment. For the employment of such a paradigm, the transaction flow is divided into three stages: Endorsement, Ordering, and Validation [3] (see figure 2.3).

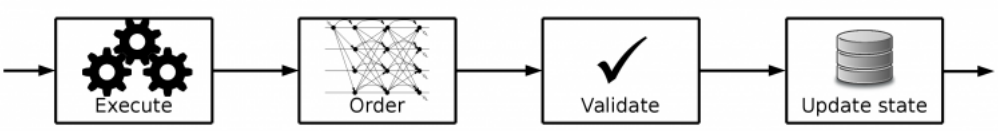


Figure 2.3: Execute-Order-Validate - Hyperledger Fabric transaction flow[3]

In the **Endorsement** phase (or execution phase), peers known as endorsing peers execute a transaction and check its correctness. An endorsement policy specifies which peers, and how many of them, need to vouch for the correct execution of a given smart contract. This allows for parallel execution of transactions and addresses possible non-determinism.

In the **Ordering** phase, the consensus protocol is employed by ordering nodes, regardless of the content of transactions, which ensures an atomic broadcast to establish order on them. The ordering of state updates is delegated to a modular component for consensus, which is stateless and logically decoupled from the peers that execute transactions and maintain the ledger [34]. Fabric also implements Gossip [35] to help with the broadcast of the ordered blocks. The gossip mechanism is scalable and agnostic to the particular implementation of the ordering service, which ensures modularity.

Finally, in the **Validation** phase, transactions are validated by application-specific trust rules (also prevents race conditions due to concurrency). The steps performed on this stage are the endorsement policy evaluation, a read-write conflict check, and the actual ledger update.

2.4.2 Blockchain in the Internet

As we are not the first to propose using a blockchain in the Internet, in the following paragraphs we describe a few solutions that propose the use of this technology in this context.

Some of the first solutions proposing a blockchain in the Internet context were as a replacement or add-on to traditional Public Key Infrastructures [36, 37]. This is an important problem, as several security incidents are related to compromised Certificate

	Guarantees/Requirements			
Existing Solutions	***Origin Authentication***	***Path Validation***	***Changes To Routers/BGP***	***Inherent Incentive Mechanisms***
RPKI [4]	**Yes**	No	**No**	No
BGPSEC [5]	**Yes**	**Yes**	Yes	No
SDX [6]	No	No	**No**	No
ROAChain [40]	**Yes**	No	**No**	**Yes**
Internet Blockchain [41]	**Yes**	**Yes**	Yes	**Yes**
BGPSECx *(this work)*	**Yes**	**Yes**	**No**	**Yes**

Table 2.1: Related proposals for improving BGP security

Authorities (CAs). These solutions thus proposed blockchains to eliminate single points of trust and to mitigate known CAs shortcomings.

Other solutions have been proposed for DNS, including Namecoin [38] and Blockstack [39]. The former provides DNS services while avoiding trust in a single entity, while the latter combines DNS and PKI capabilities. Both these solutions store name/value pairs on the blockchain, corresponding to domain names and IP addresses, respectively. These solutions are useful as they enable a censorship-resistant DNS outside the control of any single entity.

A couple of solutions have also considered using a blockchain to improve BGP security. ROAchain [40] prevents prefix and sub-prefix hijacks by storing IP prefixes and their corresponding ASes' ROA operations. This solution covers the problem of origin authentication only, so it does not addresses route manipulation attacks. The closest work to ours is the Internet Blockchain [41]. Its authors propose using a blockchain to replicate the functionality provided by RPKI, BGPSEC and DNSSEC. By using a global shared ledger, they aim to provide trust guarantees among participants, non-repudiability of transactions, and transaction history retraceability. This position paper has proposed possible transactions and discussed deployment scenarios, but does not present a specific system design nor implementation. To our knowledge, there were no follow-ups addressing this gap. In addition, its deployment model assumes that routers would have to be modified. In some sense, therefore, our work BGPSECx is the first realization of the Internet Blockchain for BGP security and, crucially, it addresses the deployment issue by running BGPSECx at SDX controllers and promoting inter-IXP collaboration.

2.5 Summary

In this chapter we provided some context for the thesis. After a short introduction to BGP, IXP, and SDX, we discussed the main BGP security issues, which together form the main motivation for this thesis. Then, we presented some of the proposed solutions to the various problems, with a discussion on their limitations. We also included a section on blockchain, one of the core technologies of BGPSECx, and a few solutions for Internet issues that propose a blockchain.

In Table 2.1 we present a comparison of our solution against selected related work.

BGPSECx is unique in providing origin authentication and path validation without requiring changes to existing routers nor to the BGP protocol. In addition, its use of a blockchain enables the introduction of incentive mechanisms for adoption.

Chapter 3

Design and Implementation

In this chapter we present the design and implementation of BGPSECx. After recapitulating the problem and motivation, we present the requirements of our solution. Then, we present the BGPSECx architecture, define its threat model, and explain its core algorithm and related data structures. Finally, we describe relevant implementation details.

3.1 Problem and Motivation

BGP is prone to two important classes of attacks [8]:

- Prefix (and sub-prefix) hijacks, where a malicious Autonomous System (AS) announces a prefix (or sub-prefix) that it does not originate.
- Route manipulation, where a malicious AS changes the AS PATH of an advertisement (e.g., a shorter path).

These attacks are usually intended to attract traffic, for eavesdropping, to create a black hole, etc. These attacks are possible in the current internet infrastructure due to two fundamental issues. The first is that there is no way to verify whether an AS announcing a certain prefix is its true owner. In other words, there is no *origin authentication*. Second, there is no way to verify the correctness of the announced paths between networks, i.e., there is no *path validation*.

3.2 Requirements

We set the following six requirements for BGPSECx:

1. Prevent prefix hijacks;
2. Prevent route manipulation attacks;
3. Full decentralization to avoid single points of trust;

4. Tamper-proofing, with no single authority able to tamper with the Origin Authentication and Path Validation repository;

5. No changes to existing infrastructure, including routers and the BGP protocol itself;

6. Capability to have incentive mechanisms to foster adoption.

3.3 BGPSECx Architecture

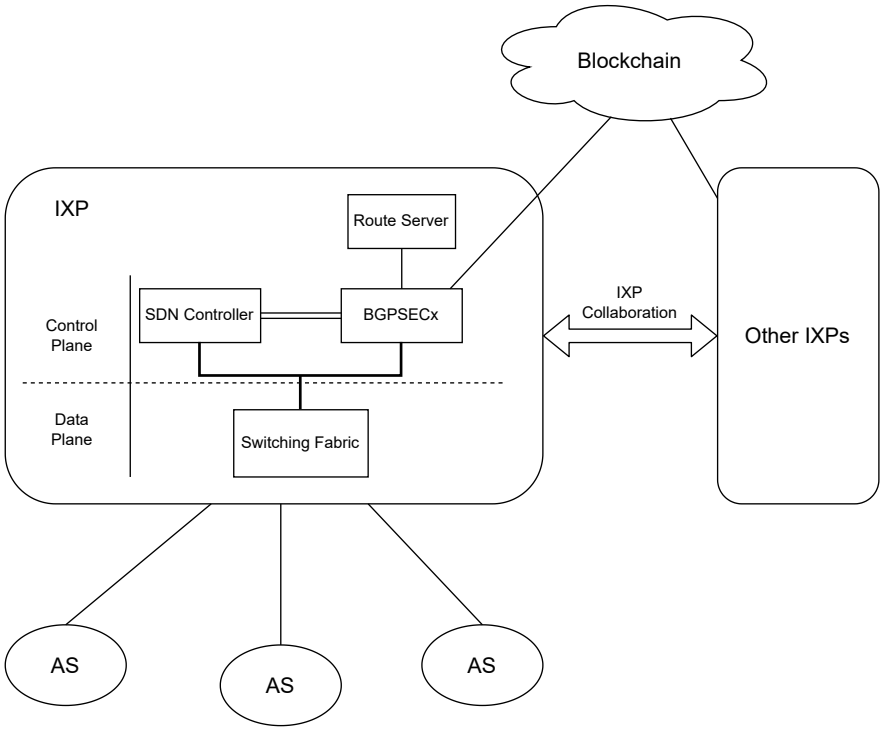


Figure 3.1: BGPSECx Architecture

BGPSECx is an SDN-based architecture that works as a secure overlay to BGP and explores inter-IXP collaboration mediated by a blockchain. This architecture is presented in Figure 3.1. We assume an SDN-based IXP, or SDX, that includes a (logically centralized) SDX controller running a route server and controlling several L2 switches. This is typical in many IXPs [6, 7].

The ASes that are part of BGPSECx negotiate a peering agreement with the IXP provider, as usual, and their routers establish a BGP connection with the SDX Route Server. We assume they use a secure tunnel (e.g., IPsec or TLS) for this communication. The BGPSECx application also runs on the controller, intercepting all BGP messages.

The BGPSECx applications running on the IXPs that are part of the BGPSECx community communicate via a secure distributed ledger: the Hyperledger Fabric blockchain in our implementation. When it receives a BGP update, BGPSECx verifies all BGP

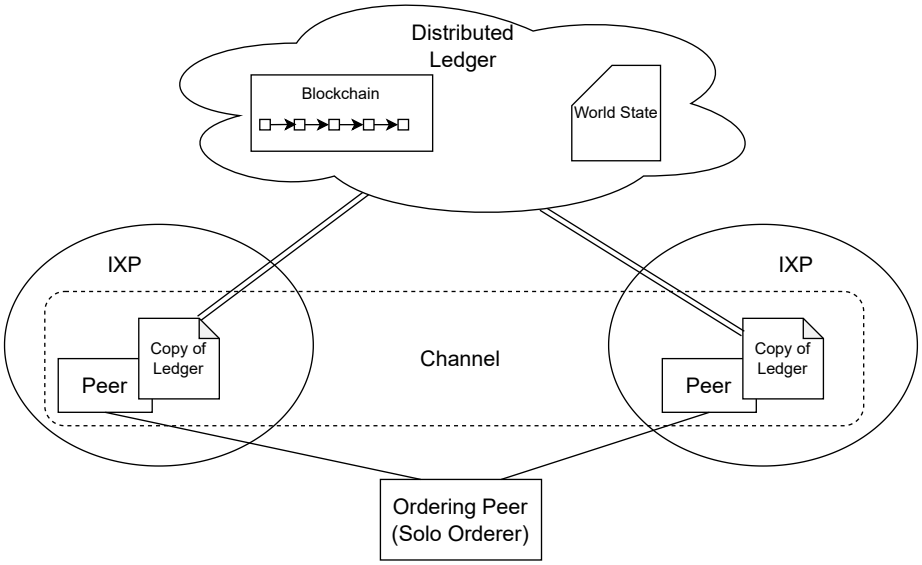


Figure 3.2: BGPSECx Chain

updates and maintains the secure ledger by means of the BGPSECx algorithm, to be described in Section 3.4.2. The update is labelled "verified" or "unverified", as will be explained later, and it is forwarded to the Route Server [10]. The BGP decision algorithm then runs, favouring "verified" over "unverified" updates. The routing and forwarding tables are then populated as usual.

The BGPSECx chain that enables inter-IXP collaboration, presented in Figure 3.2, includes one blockchain peer in each IXP. Each peer maintains a copy of the distributed ledger, and runs the BGPSECx algorithm as a smart contract. The ledger comprises two components: the actual blockchain and the world state.

The blockchain is a transaction log that records all the changes that have resulted in the current world state. Transactions are collected inside blocks that are appended to the blockchain, and they contain a history of all the changes that happened. Once written, the blockchain cannot be modified. It is immutable.

The world state contains the actual assets (in our case, the BGP Tree data structure we present in Section 3.4.1, and its associated IP Prefix). The state of the assets is expressed as key-value pairs, making it easier to directly access the current state values rather than having to traverse the entire transaction log.

3.3.1 Threat Model

In this section, we describe BGPSECx threat model, dividing it into fours parts: detail the major vulnerability of BGP; the threats that such vulnerability might pose; the attacker model; and the trustworthy components.

Vulnerability: The major vulnerability of BGP, which is the focus of this study, is the lack of security guarantees for BGP update messages, specifically, origin authentication and path validation. In BGP, a network can announce any prefix and any path, as there is no verification of the authenticity nor integrity of BGP updates. This means that the Internet routing tables can be populated with bogus paths.

Threat: The threat is the bogus announcements that disrupt the regular functioning of the Internet. The attackers can perform:

- *Prefix Hijacks*, by pretending they are the origin of IP prefixes that they do not own;
- *Path Manipulations*, by making changes to the `AS_PATH` of a legitimate BGP update.

These two situations can, in turn, lead to two different scenarios: *traffic attraction* and *traffic interception*. In the former, the manipulator's goal is to attract traffic, that is, to convince other ASes to forward traffic that is destined to the victim IP prefix towards the manipulator's own network. In the latter, the manipulator has the additional goal of ensuring that he has an available path to the victim, not to cause a black hole.

Attacker: We consider a Dolev-Yao style attacker [42], also known as an active adversary: he/she is able to log all messages and can arbitrarily delay, drop, reorder, insert, or modify messages. This attacker can therefore generate bogus BGP updates or tamper with benign ones anywhere in the Internet that is not a part of the BGPSECx community.

Assumptions: We assume that the communication channels between the BGPSECx participant AS routers and the BGPSECx SDX are secure. We also assume the channels connecting the various blockchain peers are secure, and that the Orderer service does not fail. The transactions to the blockchain have to be endorsed by a specific number of peers, as defined by the endorsement policy. A transaction is only considered correct and able to advance through the transaction flow if a majority of peers (IXPs running BGPSECx) endorse it. We assume the security of the cryptographic primitives used in all secure channels.

3.3.2 Incentive mechanisms

There are two mechanims in BGPSECx that can serve as an incentive for its deployment. First, the use of a secure distributed ledger allows any participant to verify the trustworthiness of the BGP updates sent by all other participants. This verification mechanism incentivizes honest behavior by all participants and thereby supports effective BGP security in an untrusted environment. Second, the BGPSECx chain can include a

built-in cryptocurrency, atomically coupling BGP security verification and payment in a single transaction. By including transaction fees, it can create a payment incentive for participants to add new route updates to the blockchain, thus expanding its scope, and to use it to verify BGP updates. The specifics of such payment mechanism is left as future work.

3.3.3 Rationale for BGPSECx design

The design of BGPSECx is motivated by what we consider to be the two main problems that preclude deployment of BGP security mechanisms such as BGPSEC: the cost of changing the Internet infrastructure (i.e., the routers at their core), and the lack of incentives for deployment. To address the first issue, BGPSECx leverages the data-control plane separation enabled by SDN technology, thus avoiding changes to routing infrastructure, as the core cryptographic operations run on the SDN servers. In addition, we target IXPs, due both to their central role in today's Internet topology, and the fact that many are moving to SDN-based control [6, 7]. Finally, we include a secure overlay for inter-IXP cooperation, which enables a verification process that is semantically equivalent to the BGPSEC service [5], enabling origin authentication and path validation.

Second, we use a blockchain as the core mechanism of the secure overlay and as the technology that enables the incentive mechanisms we described in Section 3.3.2. Crucially, the blockchain offers:

- decentralised trust, outside the control of any single entity, eliminating any single root of trust;
- verifiable and distributed transaction history log that is tamper-resistant, leading to transaction non-repudiability and the ability to retrace the history of any transaction – this inherently incentivizes participants not to lie about the announcements they make;
- secured by public key cryptography, with the possibility of multi-signature based authorizations for enhanced security;
- potential for a built-in cryptocurrency, atomically coupling resource transfer and payment in a single transaction, thereby enabling payment incentives, as we motivate in Section 3.3.2.

3.4 BGPSECx Chain

In this section, we describe the data structure we keep in the BGPSECx blockchain, and the algorithm we propose to update the chain and verify all BGP updates, to guarantee both origin authentication and path validation.

3.4.1 Data Structure

The Fabric world state stores a hashmap with the key-value pairs being stored. This data structure is hierarchically organized to efficiently keep BGP update records. As such, every entry to the ledger consists of an IP prefix (the key) and a *serialized tree* with the various AS paths (the value). We call this the BGP Tree. This is a simple non-binary tree that, at its root, has the first AS that announced the IP prefix (the "origin"). This AS is the "owner" of said prefix. Each child is the next AS in one of the multiple possible paths. Figure 3.3 illustrates the BGP tree of an example prefix.

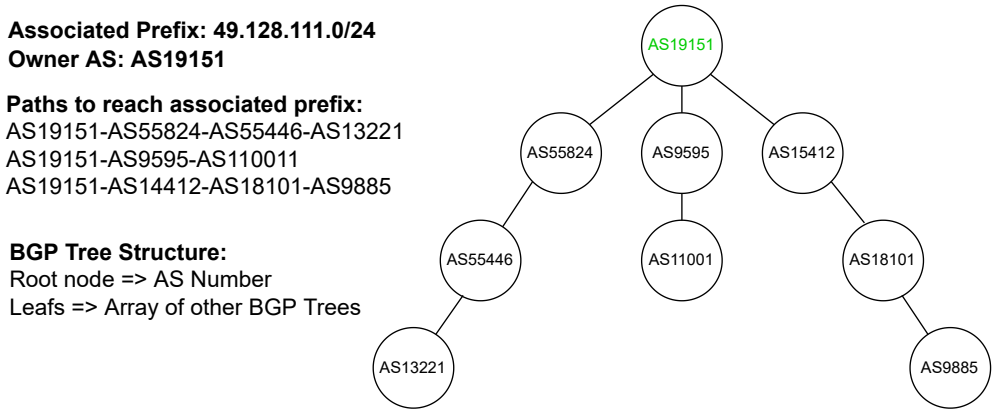


Figure 3.3: BGP tree example

With this data structure in mind, the algorithm we propose to secure BGP updates is presented in the next section. You will notice that every verification made to the blockchain uses recursive functions to traverse the BGP trees, in order to efficiently map the announced paths.

3.4.2 BGPSECx Algorithm

To better illustrate the algorithm, we present a running example in Figure 3.4. The figure shows the BGP Tree for a single prefix (1.1.1.1/24), and intends to illustrate the current state of the BGPSECx blockchain. The ASes are represented with single uppercase letters. For instance, the paths ABD and ACG were already announced and verified, and for that reason are incorporated into the blockchain.

To understand how one path gets added to the blockchain, for instance, path ABD, the following steps had to have occurred. First, AS A, a BGPSECx participant that owns IP prefix 1.1.1.1/24, has sent an update to BGPSECx with $AS\_PATH = \{A\}$. As a participant, AS A has a secure connection with the SDX that runs BGPSECx, and so this update could not be forged. In addition, the update will be stored in a blockchain, deincentivizing AS A from lying. As a result, the node A is added to the BGP Tree of this prefix.

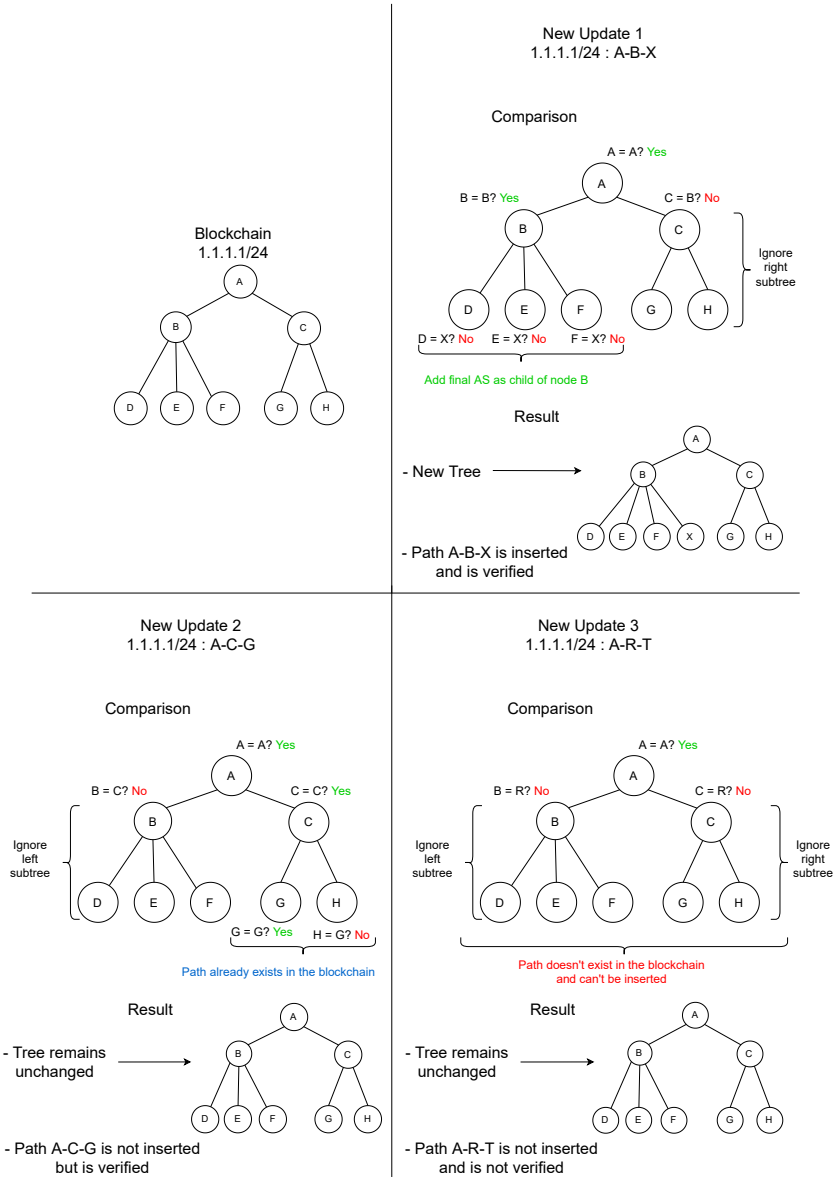


Figure 3.4: Algorithm execution with 3 example updates

Then, at a later moment, AS B, another BGPSECx community participant, sends the update $AS\_PATH = \{A, B\}$. As AS B is securely connected to BGPSECx (not necessarily in the same IXP as AS A), and path A is already in the blockchain, the new node B can be added, and hence path $\{A, B\}$ is now in the blockchain. Finally, AS D (another participant) sends update $AS\_PATH = \{A, B, D\}$ for this prefix. Since path $\{A, B\}$ is in the blockchain, this update is accepted and added.

Now consider three new updates arriving to BGPSECx, illustrating the 3 main execution possibilities. In the first example, New Update 1 (ABX), we compare A to the root node, and then to each of its children until it reaches the final AS. Since the final AS (X) is not present in the tree, and the remaining path (AB) corresponds to an existing path in the blockchain, the path is inserted into the blockchain and it is *verified*, and the tree is

updated.

In the second example, the announced path is a duplicate (ACG). In this case, the recursion keeps going down each subtree, until every AS on the path is matched to one that is already stored on the blockchain. As such, we consider that the path was *verified*, but the tree remains unchanged.

Finally, in the third example, the announced path (ART) has its first subcomponent (AR) not present in the tree. In this case, the announced update is not added to the tree, and this path is considered *unverified*.

Now we move to describing the algorithm. The algorithm starts when a BGP update is received from a participant AS. Then, the smart contract presented in Algorithm 1 is run. In the smart contract, we first consult the Blockchain to check if there is already an entry for the prefix being announced. If there is no such entry (line 27), we check whether the path length is greater than 1. If it is, then we discard that update[1]: the prefix was not in the blockchain and cannot be added because we have not yet verified its origin. If the path length is equal to 1, however, we add it to the blockchain. The reason is that this update was sent by a participant AS, so *we are guaranteed* this single AS in the update was the origin of the prefix. Recall that a participant AS is connected to BGPSECx with a secure connection, so a malicious entity cannot change it; besides, the blockchain retraceability property deincentivizes the AS from lying.

If there was already an entry for this prefix we proceed to the validation of the AS path on the announcement (starting in line 6). Since a record with the prefix in the announcement already exists in the blockchain, we compare the announced path with the stored one for that prefix. To perform this comparison, three main conditions are checked.

The first condition is if the path length is equal to 1 (line 7). This means that a BGPSECx participant AS is trying to perform an announcement as the owner of a prefix. Since this AS is securely connected, and so this update cannot be forged, this is a duplicate announcement[2] that is "verified" and can be safely discarded.

The second condition is if the path length is equal to 2 (line 10). In this case, either a path of length 2 was announced, or the recursion that goes through the announced path has reached the final two ASes. In this condition, we check if the current node we are on has any children (line 11). If it does, we have to compare them with the final AS on the announced path, to see if any of them match. Because, in this case, the announcement is either a duplicate and is discarded, or it's a different path. If it's the latter, the final AS in the path is stored as a sibling of the previously checked ASes (line 17). In both these cases, the path is *verified*. This represents the first and second cases in the running example.

[1] Note that discarding a prefix *does not mean* the update is dropped. It just means that it will not result in any update to the blockchain.

[2] In rare occasions, it could be a new announcement from a new prefix owner, but we do not treat this less common case here – we do not handle prefix withdrawals.

In the third and final condition, the announced path is greater than 2 (line 19). In this case, we have to check again if the current node has any children. If it doesn't, the announcement is discarded and the path is *unverified*, since the path cannot be longer than any other stored path by more than one AS. This is the situation reported in the third example above. If the current node does have children and its AS is equal to the first AS on the announced path (lines 23 and 24), the path is decremented by one AS and the function is called recursively against all of the node's children, in order to check the remaining ASes of the path.

To sum up, if the path is the same as one already stored, we can safely discard it, as this is a duplicate update. If the path is longer than any other stored path by more than one AS, we also discard it. This is the core aspect of the algorithm: insertions have to be incremental.

To recapitulate, to be able to add a path C of size N to the blockchain, we need to have already in the blockchain a path S of size N-1 with the exact same content of path C, except the AS that makes the announcement and that has added itself to the path.

When this process ends, the BGP update is sent to the Route Server for BGP processing, as usual. This means that the BGPSECx does not filter/drop updates, which could have potentially undesirable consequences. The BGPSECx action is therefore soft: it just labels updates as "verified" or "unverified", signalling to the BGP decision process that it can use this added information to improve the security of BGP by favouring "verified" over less trustworthy updates.

Algorithm 1 BGPSECx smart contract

```
1: B ← FabricWorldState
2: X ← newUpdate[prefix p, path c]
3: A ← announcingAS
4: V ← verificationTag
5: if p ∈ B then
6:    t ← treeStored in B(p)
7:    if length(c) = 1 then
8:       V ← Verified
9:       X is discarded
10:   if length(c) = 2 then
11:      if length(t.children) ≠ 0 then
12:         for each: l ∈ t.children do
13:            if c.tail = l.root then
14:               X is discarded
15:               V ← Verified
16:               Return V
17:         X is added to B
18:         V ← Verified
19:   else
20:      if length(t.children) = 0 then
21:         X is discarded
22:         V ← Unverified
23:      else
24:         if c.head = t.root then
25:            for each: l ∈ t.children do
26:               Return the result of this smart contract for [c.removeHead, l]
27: else
28:   if length(c) > 1 then
29:      X is discarded
30:      V ← Unverified
31:   else
32:      Xis added to B
33:      V ← Verified
34: Return V
```

3.5 Implementation

This section presents some implementation details of the solution. It specifies each of the network components, their roles, and how they are interconnected.

3.5.1 Network Components

Peers are special components of the network that allow the communication with the blockchain. There are two main types of peers:

- Endorsing Peers, who endorse transactions to be executed.
- Orderer Peers, who order the transactions into blocks.

The Orderer's function is to receive the transaction endorsements from the other peers and organize them into ordered blocks. For this thesis, we used only one ordering node, in Solo mode. This means that all block creations, as well as the order of transactions within each block, are established by this node. The reason we opted for a single such peer is simplicity: we avoid the need for a consensus algorithm, something out of scope of this work. In practice, a real-world implementation would have multiple peers working as orderers, to ensure Byzantine fault tolerance, and they would run a consensus algorithm to agree on the order of transactions for each block.

ASes are the participants of the network that produce BGP updates and are securely connected to the IXPs. The different network elements of the blockchain communicate by secure channels, to assure privacy and confidentiality to transactions. A channel is defined by members (such as the IXP organization), anchor peers (peer nodes that all other peers can discover and communicate with), the shared ledger, chaincode applications, and the ordering service nodes.

Each transaction on the network is executed on a channel, on which each party must be authenticated and authorized to transact. Each peer that joins a channel has its own identity given by a membership services provider (MSP), which authenticates each peer to its channel peers and services. For the security certificates, the Hyperledger Fabric CA is the default certificate authority component, which issues PKI-based certificates to network member organizations and their users.

Each organization (e.g., IXP) has its own Organization Certificate Authority, which issues one Root Certificate to each member (each IXP) and one Enrollment Certificate to each authorized user (the ASes who connect to the IXP). The certificate authorities create the identities that belong to each organization and issue each identity a public and private key. These keys are what allow all of the nodes and applications to sign and verify

their actions. Any identity signed by a certificate authority will be understood by other members of the network to identify the components that belong to their organization.

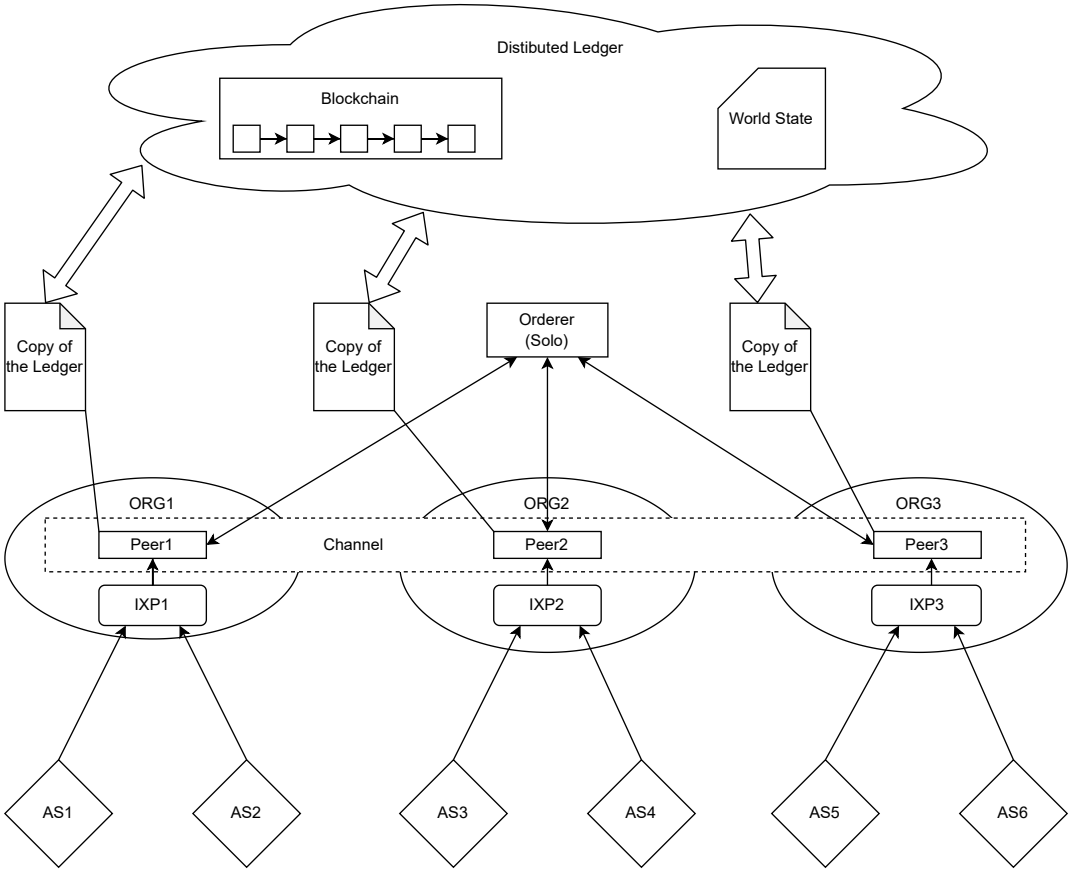


Figure 3.5: Hyperledger Fabric Network example

3.5.2 Application Logic

In this section, we describe the BGP logic that runs on the SDXs (in Java), and the smart contract that is executed by every peer, when endorsing transactions (in Golang).

BGP Logic (Java)

The application simulates the two main components of the network, the SDX at the IXP that runs BGPSECX and the ASes. ASes work as clients and provide BGP updates to BGPSECX, which then receives the updates, and processes them. The announcements are first verified in the blockchain, according to the algorithm presented in the previous section, are marked as either "verified" or "unverified", and are afterwards sent to the route server for the BGP decision process to take place.

If an update is already stored in the blockchain for that AS and the IP Prefix announced, it is considered verified. The decision process in BGP is then slightly changed to include one rule: *a verified update is preferred to an unverified one*. This represents

an additional option for the BGP best path selection process, with the goal to increase its security. Note, however, that we do not discard "unverified" updates, so we do not affect the normal BGP update process. We just prefer, when available, "verified" routes.

Smart Contract (Golang)

Specific APIs to access the blockchain data allow for queries and invokes (regular database reads/writes). Functions may require, among other arguments, a key to access the values stored in the ledger (a regular map with key/value pairs). The keys are the IP prefixes and the values are the AS paths associated with the updates.

For specific queries (to read what is stored) or updates (to add or modify stored information), the IP prefix that is to be accessed needs to be passed to the function as an argument, which then returns a serialized data structure (called the BGP Tree, as previously explained) that contains all verified paths of the announced prefix. The serialized BGP Tree is then passed to a specific function that deserializes it to a BGP Tree object, which can then be accessed for reads and writes.

The blockchain logic is applied here, with the AS_PATHs in the updates being checked to see if they should be added to the blockchain (as they are considered "verified") or if they should be left as "unverified".

3.6 Summary

This chapter presented the design and implementation of BGPSECx. We started by motivating the problem, and presented the requirements and the architecture of the proposed solution. We detailed the threat model, the incentive mechanisms, and presented the rationale for our design. Then, we described the data structure kept in the BGPSECx blockchain, and the algorithm we proposed to update it and to verify BGP updates, the core element of our approach to enable origin authentication and path validation. Finally, we presented some implementation details.

In the next chapter we evaluate our solution.

Chapter 4

Evaluation

The main objective of BGPSECx is to prevent prefix hijack and path manipulation attacks to BGP. For evaluation we will simulate such attacks in a quantifiable way, to understand how many malicious BGP updates could be detected and potentially mitigated.

We divide the evaluation in two parts: the effectiveness of our proposal, using realistic BGP data, and system performance of the prototype developed in this thesis. We will start each part with a description of the methodology, and then present and discuss its main results.

4.1 Effectiveness Evaluation Methodology

In this section, we explain the evaluation methodology for BGPSECx effectiveness. We will detail the setup, the metrics to be evaluated, and the datasets used.

4.1.1 Datasets

We start with real BGP traces from CAIDA datasets[1]. We thus assume the BGP updates in these traces to represent the BGPSECx BGP traffic. Each individual dataset represents a period of 5 minutes throughout an entire day, totalling 288 traces for one day. This represents one execution, and there will be a total of 10 executions, with different datasets used each time. For this work, only the announced IP prefix and AS path of a BGP update were relevant for the data-driven simulations, so we made some data cleaning to improve simulation speed.

The evaluation consists of two phases, Training and Testing. The Training phase will provide the application 75% of the daily individual datasets (216), containing updates that are deemed correct ("non-malicious"), and their purpose is to "train" the application, by populating the blockchain with BGP announcements, following our algorithm (Section 3.4.2).

[1] CAIDA.org collects several different types of data, including real BGP update traces, at geographically and topologically diverse locations, and makes this data available to the research community.

The Testing phase will consist of the remaining 25% datasets (72), and these will have, among the regular BGP updates, some synthetically generated "malicious updates"[2], their purpose being to test the application algorithm and its effectiveness.

In the Testing phase, the "malicious updates" that were introduced into the datasets can be divided into two types of attacks on BGP: prefix hijacks and path manipulations. For every 1000 entries, on average (with 10% deviation), we input a malicious update. These entries are generated with random IP prefixes gathered from the datasets from the Training phase. In total we inject 100 malicious entries: 50 prefix hijacks, and 50 path manipulations.

For each type of attack, the malicious entry would have a "malicious AS" inserted somewhere in the path, referred as "ASXXX". For prefix hijacks [44], the "malicious AS" is inserted at the beginning of the path: this AS pretends to be the owner of the prefix that it was announcing. For path manipulations [44], the "malicious AS" is inserted anywhere in the path, except in the first and last positions[3]. Below we present two example malicious updates:

- 1.1.1.1/32 : ASXXX ->AS1 ->AS2 ->AS3 ->AS4 ->AS5 (prefix hijack),

  or

- 1.1.1.1/32 : AS1 ->AS2 ->AS3 ->ASXXX ->AS4 ->AS5 (path manipulation),

where "1.1.1.1/32" is the prefix that is being announced, "ASXXX" is the malicious AS, and all the others are regular ASes in the path (present in the training dataset).

4.1.2 Evaluation setup

We perform 10 runs of the simulations, using 288 datasets in each run. These 10 runs then represent 10 different days. Additionally, we will consider as a metric a certain number of ASes on the announced AS paths to be "trusted", i.e., we assume they are part of the BGPSECx community. We will consider a total of 1, 2, and 3 ASes to be trusted, meaning 10 different runs for each, bringing the final number of runs to 30.

The metric **number of trusted ASes** corresponds to the size of the paths that the algorithm will initially accept as "verified". To make this clear, ideally, for any prefix, we would only accept announcements with one AS in the path as "verified", which would represent the owner of the prefix, in the warm-up phase of the execution. This is what best simulates our model, as we assume only the last AS in the PATH to be directly connected to BGPSECx. However, we slightly extend this model to understand the gain of extending the scope of the BGPSECx community.

[2]Unfortunately there is no public data with labeled hijack and/or path manipulation events [43].

[3]The first would be a prefix hijack; the last would be accepted as "benign" in our model as we assume the last AS in the path to be a participant – recall a participant is directly connected to BGPSECx by a secure connection.

Note that the number of trusted paths has an influece on the testing dataset. While a malicious entry can have a random number of ASes on the path, they are always greater than the number of ASes considered as "trusted".

The Training phase consists of each BGP update to trigger the BGPSECx algorithm verifications to decide whether the update should be added to the blockchain. Besides this, the Testing phase performs additional verifications to determine if each update is "verifed" or not. There are thus two metrics of interest: the **number of updates inserted** into the blockchain, and the **number of updates verified** by BGPSECx.

4.2 Effectiveness Evaluation Results

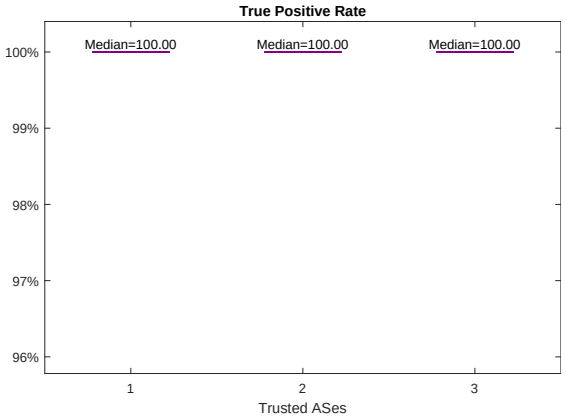
To evaluate the effectiveness of BGPSECx we computed the following values:

- **True Positives, TP**, which corresponds to **malicious** BGP updates marked as **unverified** and prevented from entering the blockchain.
- **True Negatives, TN**, which are **non-malicious** updates that were marked as **verified** and were potentially stored in the blockchain (we say potentially because in case of repeated updates they do not get stored again).
- **False Positives, FP**, which corresponds to **non-malicious** updates that the algorithm marked as **unverified**, thus also prevented from being stored in the blockchain. In this case, not enough knowledge was gathered in the blockchain to verify that this update was in fact benign.
- **False Negatives, FN**, are **malicious** BGP updates marked as **verified** and thus were potentially added to the blockchain. This is particularly undesirable as these malicious updates will pollute the BGPSECx chain.

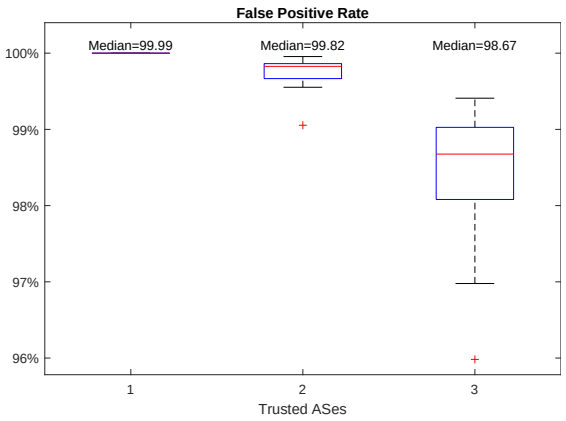
From these, we calculate the following metrics to assess our solution:

- **True Positive Rate**, the ratio of malicious updates prevented to enter the blockchain (TP) over the total number of malicious updates (TP+FN). This metric allows reasoning about the number of malicious updates that pollute the BGPSECx chain.
- **False Positive Rate**, the ratio of benign updates prevented to enter the blockchain (FP) over the total number of benign updates (TN+FP). This metric allows reasoning about the growth of the BGPSECx chain – which we can asses with a proxy: increasing the number of trusted ASes. As more benign updates are added to the blockchain, we should expect a higher number of verified updates, and hence a higher probability of avoiding BGP attacks.

In Figure 4.1 we present two plots with the results that summarize all experiments, one for the True Positive Rate (TPR), and the other for the False Negative Rate (FNR). The box plots represent the aggregated values over all executions, and considering only 1, 2, or 3 ASes as trusted. On these, the vertical axis represents the TPR and FPR, whereas the horizontal axis differentiates the number of ASes that were initially trusted, ranging from 1 to 3, as previously explained.



(a) TPR for 3 scenarios: 1, 2, and 3 Trusted ASes. **Higher is better.**



(b) FNR for 3 scenarios: 1, 2, and 3 Trusted ASes. **Lower is better.**

Figure 4.1: Measuring the effectiveness of BGPSECx with the True Positive Rate (TPR) and False Negative Rate (FNR)

We take two main conclusions from these results. First, the TPR (Figure 4.1a) is equal to 100%: there are no False Negatives. This is a relevant point, as our solution guarantees that *no malicious updates pollute the* BGPSECx *chain*. But it should come to no surprise, given the logic of our algorithm. The reason is that *all* the updates added to the blockchain are generated *by* BGPSECx *participants*. They are thus directly and securely connected

with BGPSECx, and so it is guaranteed *that the last AS* of that update's AS_PATH is correct. In addition, as an update is added to the blockchain only if the previous ASes in the AS_PATH were all verified before and thus are in the blockchain already (recall Section 3.4.2, and Figure 3.4 in particular), we are assured the full path was verified. Note also that these updates are in general generated by participant ASes that will often be connected to different BGPSECx IXPs. Importantly, the secure overlay between IXPs guarantees that no malicious entity can tamper with the BGPSECx chain. It should be stressed that this procedure directly mimics the incremental, AS-by-AS verification mechanism of BGPSEC [5], and as such is semantically equivalent to it.

Second, the FPR (Figure 4.1b) is also close to 100%, as the number of False Positives is high compared to the very small number of updates effectively malicious. In our datasets, this number is indeed low[4]: only 100 in over 12+ million updates (check Figure 5.1 in the Appendix for instance). Recall that False Positives correspond to updates that are marked as "unverified". Put in other words, we are unsure whether they are benign or malicious, and so do not improve over the current situation. The BGPSECx blockchain does not yet contain enough information to help us "verify" these updates.

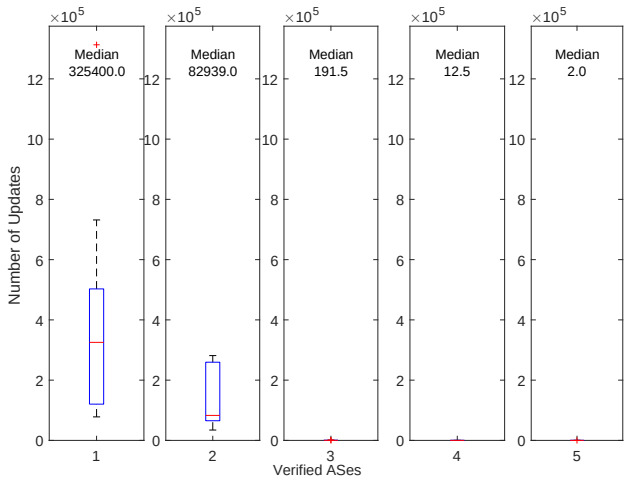
That is why it is relevant to consider the three cases presented in Figure 4.1b. As we increase the number of Trusted ASes, and thus BGPSECx scope, we see a clear reduction in the FPR. The reason is that the blockchain has more information added, resulting in a sharp increase in the number of True Negatives (benign updates that we can verify with BGPSECx), from the few hundreds to the several tens of thousands (e.g., compare Figure 5.1 with 1 Trusted AS with Figure 5.3 with 3 Trusted ASes in the Appendix).

Finally, we should note that for computational and time constrains our traces cover only a 24-hour period. We should expect the benefit of a system such as BGPSECx to increase not only as the number of participant ASes increases, but also after several months in production. We leave it as future work to evaluate the system on larger time periods (months).

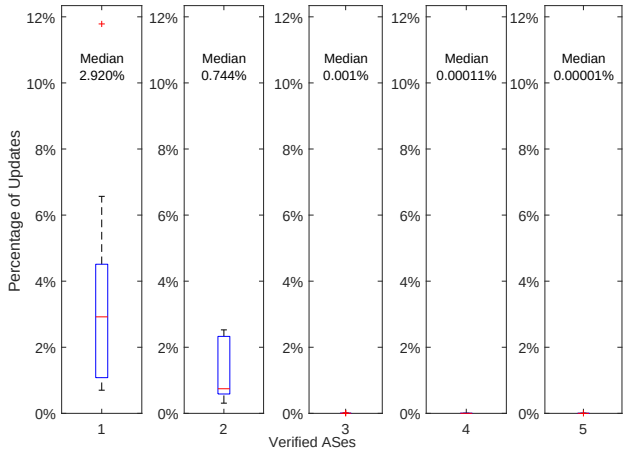
Partial verification. Up until now, we have considered that an update was considered as either "verified" or "unverified", as a discrete (0 or 1, "good" or "not sure") signal to the subsequent BGP decision process. In the next, final experiment we consider the possibility of sending a more fine-grained, "continuous" signal to assist the BGP process. For this purpose, we introduce the concept of "partial verification" of an announcement. For a certain prefix, if an update has part of its AS_PATH verified in the blockchain, but not its entirety, we consider that update to be **Partially Verified**. This may be useful as having a subset of the update verified, instead of none, arguably increases the security of the process.

Figure 4.2 presents the results (both absolute and relative values) considering the sce-

[4]We decided to be conservative, but we are supported by empirical evidence that the fraction of malicious updates, while concerning, is relatively small [43].



(a) Partially verified updates (absolute values)



(b) Partially verified updates (percentages)

Figure 4.2: Partial verification of BGP updates with 3 ASes initially trusted

nario with 3 Trusted ASes. We chose this scenario given its larger scope, as it makes it clearer the potential value of partial updates. In the Appendix we also present results for the other two scenarios (1 and 2 Trusted ASes). As can be seen, it was possible to partially verify BGP updates with AS_PATHs that are up to 5 ASes long. In our 1-day long datasets BGPSECx was able to verify the origin of over 325k updates (i.e., the first AS was verified), around 3% of the total updates, and thus extend the ability to prevent prefix hijacks. Other 80k+ updates would also see both the origin AS and the previous AS in the path verified (a guarantee equivalent to Path-End Validation [45]).

4.3 Performance Evaluation Methodology

This section covers the methodology for the evaluation of the performance of our proposed solution. It provides a description of the more simplified datasets that were utilized, as well as the larger degree of preparation required for these tests. It also goes into detail about the network topology that was necessary in order to represent a real-world scenario and the correspondent testbed.

4.3.1 Datasets

For the performance evaluation, a different collection of datasets were utilized. They were still based on real BGP updates, however, since the point was to test the solution utilizing a real blockchain and how it performed under heavy loads, the datasets were synthetically generated. The generations program ensured that the announcements would be incremental and ordered. What this means is that they would have various BGP updates with single-AS path announcements and that the subsequent updates would increment them, one AS at a time. This way, it is guaranteed that the algorithm would accept a large number of updates into the blockchain. An example of this would be:

- 1.1.1.1/32 : AS1
- 1.1.1.1/32 : AS1 ->AS2
- 1.1.1.1/32 : AS1 ->AS2 ->AS3
- 1.1.1.1/32 : AS1 ->AS2 ->AS3 ->AS4
- 1.1.1.1/32 : AS1 ->AS2 ->AS3 ->AS4 ->AS5

where, for the prefix “1.1.1.1/32”, there would be five announcements, each with an increasing number of ASes, in an orderly fashion.

For simplicity, all the datasets were created with 1000 prefixes, which averaged on somewhere around 5000 entries per dataset (when counting the generated number of entries based on the random number of ASes for each path). After the datasets were entirely created, they were sorted by path length, so as to not have all of the entries for the same prefix clumped together, which would cause insertion conflicts due to the MVCC, or Multi-Version Concurrency Control. The MVCC is a mechanism implemented in the Hyperledger Fabric blockchain that prevents concurrent updates of the same key.

4.3.2 Evaluation setup

To evaluate the efficiency and scalability of the solution, two important metrics were taken into consideration. The first was **latency**. Not only of the total execution of the

program but also of each individual transaction. For this, timestamps were measured whenever a transaction started and as soon as it ended. A transaction is defined as terminated whenever the blockchain emits the event stating that said transaction has been committed. In our experiment, a transaction corresponds to one BGP update (i.e., one entry from the datasets).

Timestamps for when the program started and terminated were also measured. This allowed us to calculate the latency of the entire program (i.e., the full dataset was completed). There was a specific thread that served all the requests, and the timers were placed at the beginning and the end of it. This way, it is possible to verify how the program behaves under different loads and with different network topologies.

The second metric was **throughput**. By dividing the total number of transactions on each execution, by the total time it took to finish them, it is possible to define the maximum throughput of the proposed implementation. This is the most important figure of merit in defining if such a solution is feasible in a real world scenario.

For these experiments, each dataset was executed 5 times, in order to guarantee that most of the BGP updates were accepted into the blockchain. The few that timed out in the earlier runs ended up being accepted in the later ones. After 5 executions, no more updates would be stored in the blockchain, even if the datasets were executed continuously, thus terminating the experiment. There were 4 different datasets, which were run 5 times each, bringing the total of executions to 20. After each dataset was executed 5 times and the values were measured, the blockchain was reset and the process would start again with the next dataset.

The design of the solution requires that, for every BGP update that tries to be committed to the blockchain, a comparison be made between the IP prefix of the new update and the IP prefixes that are already stored. We do this for the algorithm to decide if the prefix should be inserted as a new one, if it is a sub-prefix of one already stored and must update it, or if it is an extension of a stored one (a prefix that encompasses it) and must be discarded.

With this in mind, it is easy to see that as the blockchain gets populated with different prefixes, the amount of processing required for each new prefix also increases. However, the major problem here comes from the fact that different transactions accessing multiple different keys might trigger a behavior specific to the Hyperledger Fabric framework known as a Phantom Read.

Specifically, during the validations and commit of transactions on Fabric, each peer in the channel will validate each transaction in the block to ensure it has been endorsed by the required organization's peers, that its endorsements match, and that it has not become invalidated by other recently committed transactions, which may have been in-flight when the transaction was originally endorsed. Here is where the problem arises. In order to check that a transaction has not been invalidated by another one, in this phase, the peers

invoke the chaincode again and compare the read/write sets that they get with the ones from the endorsement phase.

By having to compare the prefix from one transaction with many from the blockchain, the read sets tend to overlap. As the blockchain gets more populated, it is more likely that one transaction makes a query to a prefix that ends up being altered before it gets to the commit phase. This means the read set it gets from the endorsement phase and the one it gets from the commit phase are different. This is known as a phantom read. The transactions might not even affect the same prefixes, but the simple fact that the ranged query is performing a read on a prefix that is changed between endorsement and commit time is enough to trigger this behavior. And if a phantom read occurs, that transaction is invalidated. This, in turn, means that the more prefixes the blockchain has, the more likely it is to have "failed" transactions inside a block.

Based on this knowledge, an additional performance metric was introduced in the tests, which is the **number of valid transactions per block**, in order to understand how they evolve over time. By checking how many transactions are valid within a block, it is possible to see a clear connection between the number of blocks that have already been inserted and the transactions that have failed. Furthermore, this helps us better understand how viable this solution is for a real scenario.

4.3.3 Network Topology

In order to have the tests yield meaningful results, it was necessary to simulate a network with all the required participants for a project of this scope. As such, the tests were run on a network that comprises 4 Organizations (each one on its own physical machine). For each of them, there was one Certificate Authority Organization (which deals with the issuing of all of the cryptographic material for the components and users of the blockchain), one CLI (command-line interface, which is used to issue direct commands on the peers), and two peers. Aside from all of this, one of the machines also had one Orderer (for testing purposes, only one orderer was used in Solo mode). All of these "participants" ran on their own container using Docker.

For the actual testing of the project, each Organization had one IXP and two ASes. For simplicity's sake, the IXPs and the ASes work as servers and clients, respectively, with the clients sending their requests (the BGP updates), to the IXP for it to add them to the routing table, and then send them to the peers, in order to be added to the blockchain.

Figure 4.3 presents the main components of the network topology utilized for the testing, as well as how these connect. For simplicity, the Certificate Authorities and the CLIs were ommited.

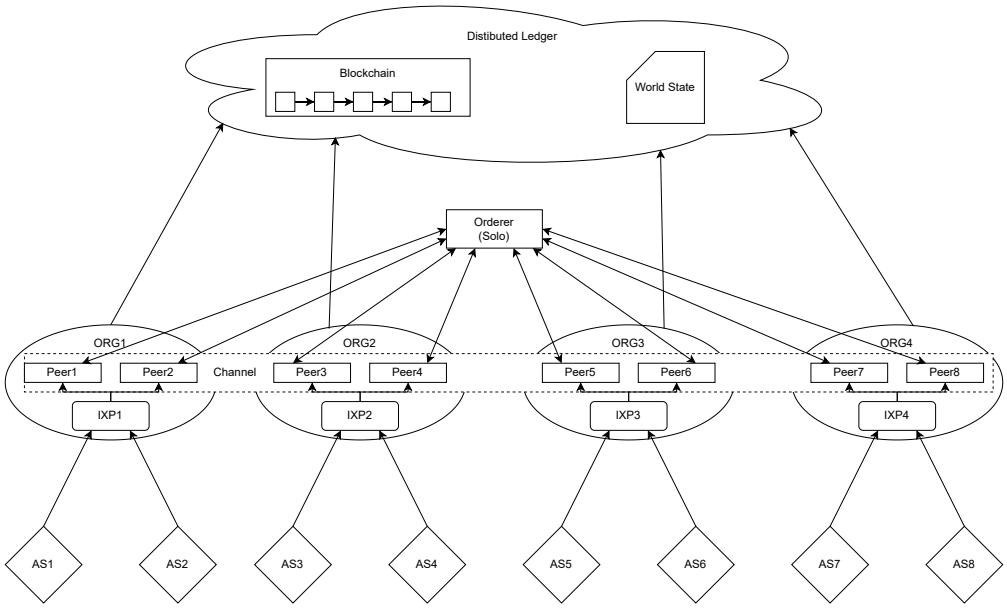


Figure 4.3: Hyperledger Fabric network topology for performance testing

4.3.4 Testbed

The physical machines that were utilized for the testing were the machines in FCUL's Quinta, which is one of LASIGE's processor farm (computational cluster dedicated to large-scale experiments in distributed systems). More specifically, the machines used were from cluster S, which comprises 16 Dell PowerEdge R410 nodes, with 2 Intel Xeon E5520 CPUs (2.27 GHz / 1 MB L2 cache / 8 MB L3 cache) each, four cores per CPU and two threads per core. It also had 32 GB (8x4GB) / DIMM Synchronous 1066 MHz (0.9 ns) of memory. Each of the 4 machines used ran Ubuntu 20.04 (64-bit).

4.4 Performance Results

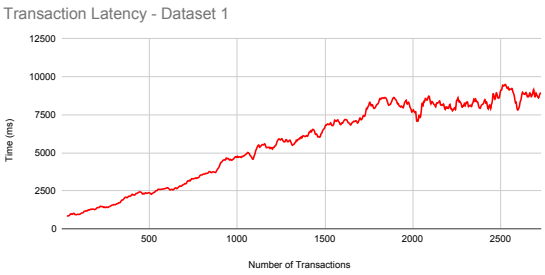
To evaluate the performance of the solution, a few measurements were taken. For the latency, we measured the total time a transaction takes to finish, from the creation of its specific thread to the completion of the whole process (transaction emitted event being received on the client-side). We also measured the total latency of the entire run, from the moment the IXP started to receive BGP updates until it finished the last transaction. Based on these values, it was calculated the transaction throughput of the blockchain.

Each of the four graphs in figure 4.4 corresponds to a test with a different dataset. On each test, the dataset was executed 5 times in a row, in order to guarantee that the maximum amount of BGP updates would be inserted into the blockchain.

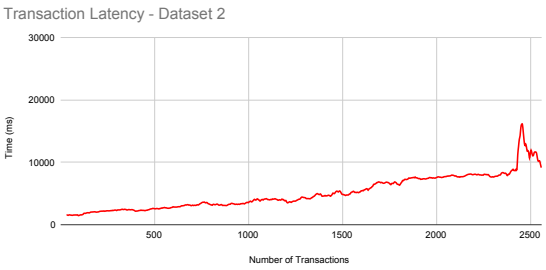
As the test results show in table 4.1, we can see that the average throughput is around 13 transactions per second, with a latency of at most 11 seconds (average 4.9 seconds), albeit with a few outliers (Fig. 4.4). These results were obtained with a workload of between

Datasets	**1**	**2**	**3**	**4**	**Total**
Throughput (tps)	12,42	12,29	14,14	11,95	12,70

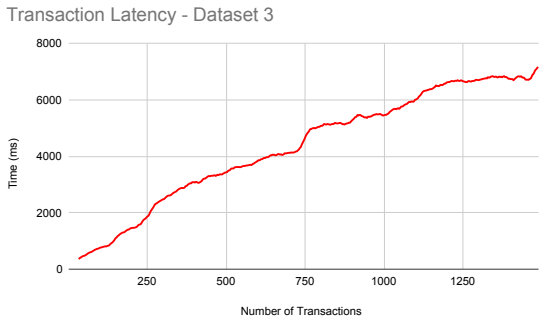
Table 4.1: Transaction throughput (Transactions per second) using different datasets



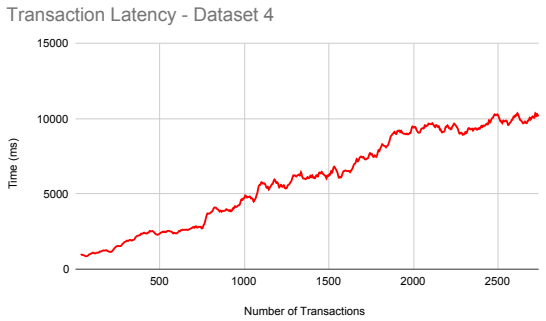
(a) Dataset 1



(b) Dataset 2



(c) Dataset 3



(d) Dataset 4

Figure 4.4: Latency of transactions using different datasets

1500 and 2800 transactions in parallel. The graphs in Figure 4.4 show an almost linear growth when it comes to transaction latency. At a couple hundred transactions, latency is below 2 seconds, which could be considered acceptable. However, as the transactions increase and the blockchain gets busier, so does the latency, rising to slightly above 10 seconds.

The previously mentioned outliers, whether they be drops or spikes, could be explained by repeated BGP updates and near-timeouts, respectively (see below). The files used between executions were the same, which means eventually some updates will be repeated. Transactions with repeated BGP updates are bound to be discarded by the peer that is performing the proposal because it does not satisfy the requirements defined by the BGPSECx smart contract. This means it does not even get to the Orderer to be put in a block and sent to the peers, which is where the greatest bottleneck is, thus having a very small latency.

Near-timeouts are also regular occurrences of the framework, most likely due to a

bug in the implementation, and they usually occur under heavy loads. The peers lose communication and the transactions get "stuck" until they eventually fail due to timeout. In the project, the timeout was set to 30 seconds, in order to better understand this specific behavior. Because of this, some transactions are able to complete even with absurdly high latency values.

Nevertheless, even with the increment in proposal wait time, it can be seen that, of the 6000 transactions that are being provided to the application, only at most 2800 are going through (Fig. 4.4). It is unclear whether this is due to the transaction processing surpassing the defined 30-second threshold, as previously mentioned, or if it is indeed due to a bug in the implementation of Hyperledger Fabric that makes some transactions fail abruptly. As for the throughput, 13 transactions per second is not very high, however, it is also not unexpected, considering the amount of processing required for each commit to the blockchain.

It should be noted that these values could be increased, if the network had more endorsing peers per Organization (as a way to better balance the load) and a better ordering service. When looking at the BGP Instability Report [46], we can see that, based on the traffic from the 50 most active ASes, the average frequency of BGP update messages per second is 13.39. This is on par with the average throughput of Hyperledger Fabric (under heavy loads), which means that HLF could deal with the current BGP requirements. It should also be noted that the performance values taken from the application tests were when using IPv4 and IPv6 prefixes in conjunction, whereas the tests from the BGP Instability Report considered only the former. Since IPv6s are heavier when it comes to the processing required, if the datasets used for the application only had IPv4s, we would probably achieve better results, i.e., higher throughputs.

The biggest performance bottleneck was the fact that for block ordering, it was used the Solo Ordering service, which utilizes only one ordering node. It is this node that determines the order of the transactions in the blocks and sends them to the peers for the committing phase. The remaining peers from the other organizations also play a significant role, since they are required to endorse the transactions that the "working peers" are emitting. The endorsement policy utilized was very simple, requiring just that, for every transaction, one peer of each organization endorse the transaction. If for any reason, a more strict policy were required, based on the application's needs, it would probably increase the transaction latency, since it would increase the communication between peers. Thus, the endorsement policy utilized was the most efficient for this project.

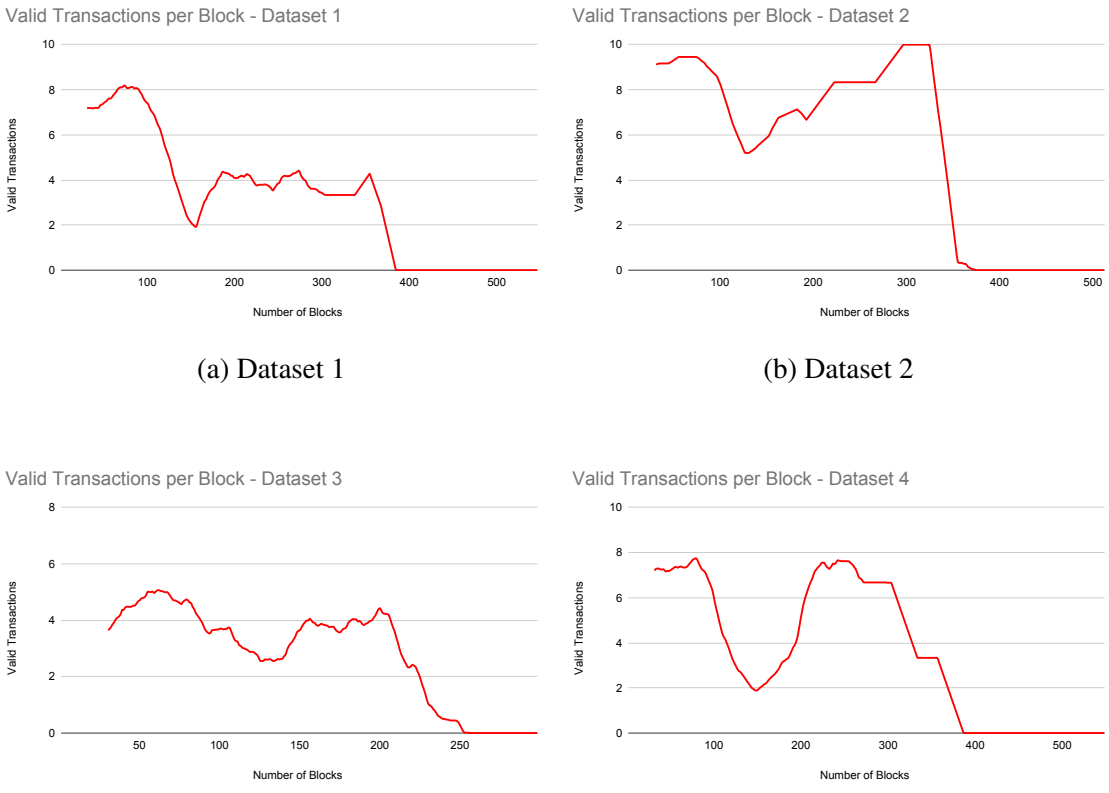
As we can see in Melo et al. 2019 [47], when using the Solo Ordering service, under light workloads (600 clients, which for the purposes of this thesis, can be comparable to 600 transactions), the blockchain reaches a throughput of around 300 transactions per second, with a latency of 1 second. These values go down significantly if the workload doubles to 1200 clients, with throughput coming down to 260 transactions per second,

and latency reaching 12 seconds. When comparing the project of this thesis with the one from the mentioned paper, it is possible to see the significant difference in performance (although if the workload on the paper were to be increased to numbers similar to the ones tested on this project, the results could have been closer).

This discrepancy in terms of throughput can be explained by the intense processing each of the transactions in this project requires. The fact that each transaction performs a ranged query, inspecting multiple prefixes in order to find the one it must alter, increases the processing significantly, especially when considering that the space of prefixes tends to increase over time, thus increasing each subsequent ranged query.

If, on the other hand, we were to compare it to the Bitcoin blockchain (Bitcoin has a fixed throughput of 7 transactions per second and average latency of 8 minutes) [48], it could be said that there is a slight improvement.

Finally, in Figure 4.5 we present the number of valid transactions per block. As previously explained, transactions that query the same prefixes trigger phantom reads, which invalidate transactions, even if they alter distinct prefixes. This metric was added to evaluate the number of transactions that are invalidated as the blockchain gets more populated with BGP announcements.



(a) Dataset 1

(b) Dataset 2

(c) Dataset 3

(d) Dataset 4

Figure 4.5: Valid transactions per block using different datasets

Even though the transactions in the blocks might not be in the order they were an-

nounced (since it is up to the Orderer to define in which block a transaction gets stored), there are two things that can be seen from these plots. The first is that, despite some blocks having the maximum number of transactions valid (10), there are still many in between with very few valid transactions (less than 3 transactions per block, or even 0 sometimes), which means that there is not a specific reason why they fail, other than the fact that they were ordered in a way that prevented them from being accepted into the blockchain (and altering the world state). This introduces a certain degree of randomness to the process that cannot be predicted.

Also, it should be noted that the transactions in the blocks can be both queries and updates to the world state, which means that some of the blocks that have 10 valid transactions may very well be just queries, hence their validity. Queries are not affected by the Phantom Read problem, because this mechanism only invalidates transactions that actually alter the world state. Thus, queries are always valid and updates can fail arbitrarily.

The second observation, and possibly the worst when it comes to the performance of the implemented solution, is that the perceived tendency is for the valid transactions in the blocks to keep dropping with time until they reach 0. This means that the bigger the blockchain is and the longer the updates keep coming, the fewer the transactions that will actually be valid and update the blockchain.

4.5 Summary

In this chapter we evaluated BGPSECx, our proposed architecture to improve the security of BGP. We have shown that the solution can be effective, and that its effectiveness can dramatically increase as one broadens the scope of the system – i.e., as more ASes join the BGPSECx community. While still limited, the performance of our preliminary prototype already achieves a performance that may be considered reasonable, as it is not far from related systems, and it is slightly better than the Bitcoin blockchain. But there is still plenty of room for optimizations to improve both system throughput and latency.

Chapter 5

Conclusion

In this thesis we proposed a new architecture to improve BGP security. Our proposal, BGPSECx, uses SDN technology and explores inter-IXP collaboration using a secure overlay mediated by a blockchain. These characteristics allow achieving three goals: origin authentication and path validation of BGP updates (the same guarantees offered by BGPSEC), avoiding changes to routers (cryptographic operations are performed in SDN servers), and offering incentives for honest behaviour by the participants (via the blockchain).

The BGPSECx algorithm's insight is that it follows an idea that is equivalent to BGPSEC, where each AS digitally signs each of its BGP messages, with the signature of an AS also covering all the signed messages received from the previous ASes on the path. Our twist is that we offload the cryptographic operations to the SDN controller that connects to other SDN controllers from the BGPSECx chain via a secure overlay, thus avoiding changes to routers and online cryptography, which could require costly hardware crypto acceleration.

We believe there are two aspects of BGPSECx that are crucial for deployment. First, we target deployment at IXPs, which are playing an increasingly central role not only in the Internet topology but also in offering added value services to its participant ASes. For instance, many peer ASes now use the Route Server at IXPs [10] to establish connections to other peers, and some even use prefix filtering mechanisms using the RPKI. Our idea is to increase IXP offerings with BGPSECx, further extending the surface of origin authentication and also offering path validation, for the first time. Second, the use of a Blockchain as mediator with its distributed trust guarantees, its tamper-proof characteristics, and the possibility to integrate a currency, de-incentive "bad" behaviour and can enable economic incentives for adoption.

We envisage several avenues for future work. For the short term, we intend to increase the scope of the evaluation to larger datasets that cover a time period of months or years. We are also working to obtain BGP datasets from various IXPs to perform trace-driven simulations in a more realistic inter-IXP collaboration setting. In parallel, we are inves-

tigating the use of recently proposed solutions (e.g., [43]) that track malicious updates, as a way to label BGP traces with this information, and thus avoid the need for synthetic malicious updates. Future work also includes devising a crypto-currency mechanism as the one we motivated in this thesis to create economic incentives for BGPSECx adoption. Finally, we are working on optimisations for our prototype, namely to the distributed ledger component.

Appendix

Data on Total BGP Updates

The following graphs contain the raw date collected throughout the effectiveness tests. Each graph represents one experiment with a single dataset, considering either 1, 2, or 3 ASes as initially trusted by the algorithm.

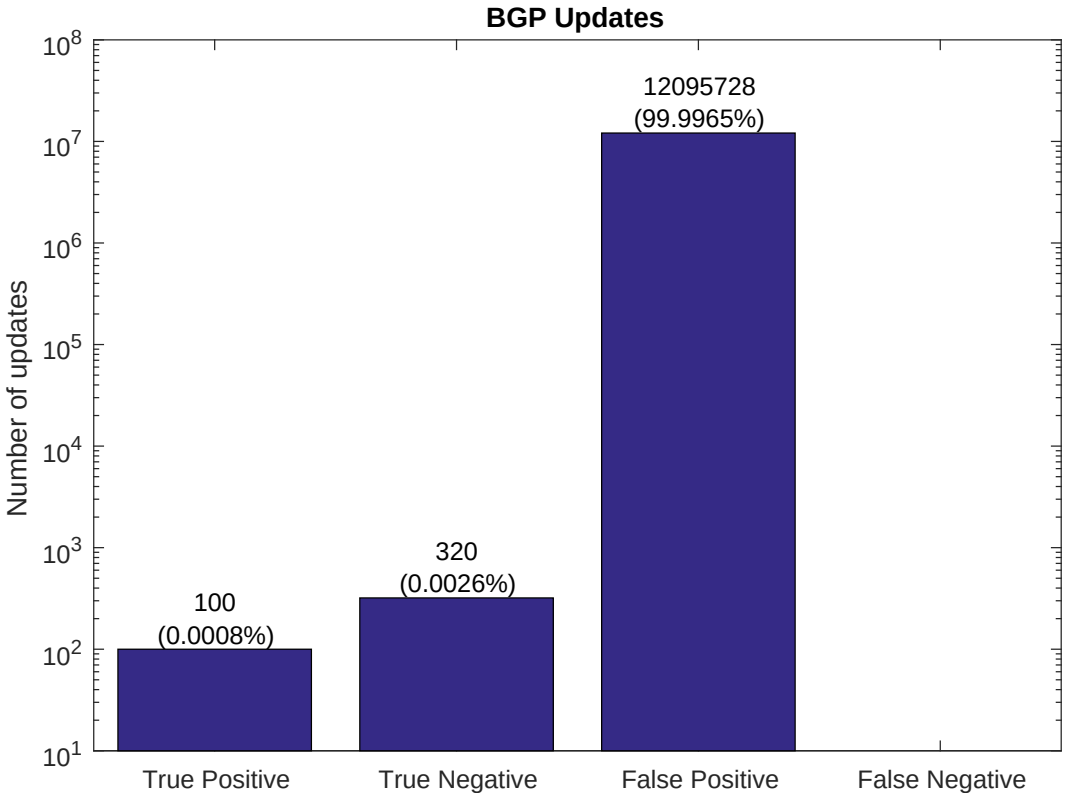


Figure 5.1: BGP Updates - Dataset 1 - 1 Trusted AS

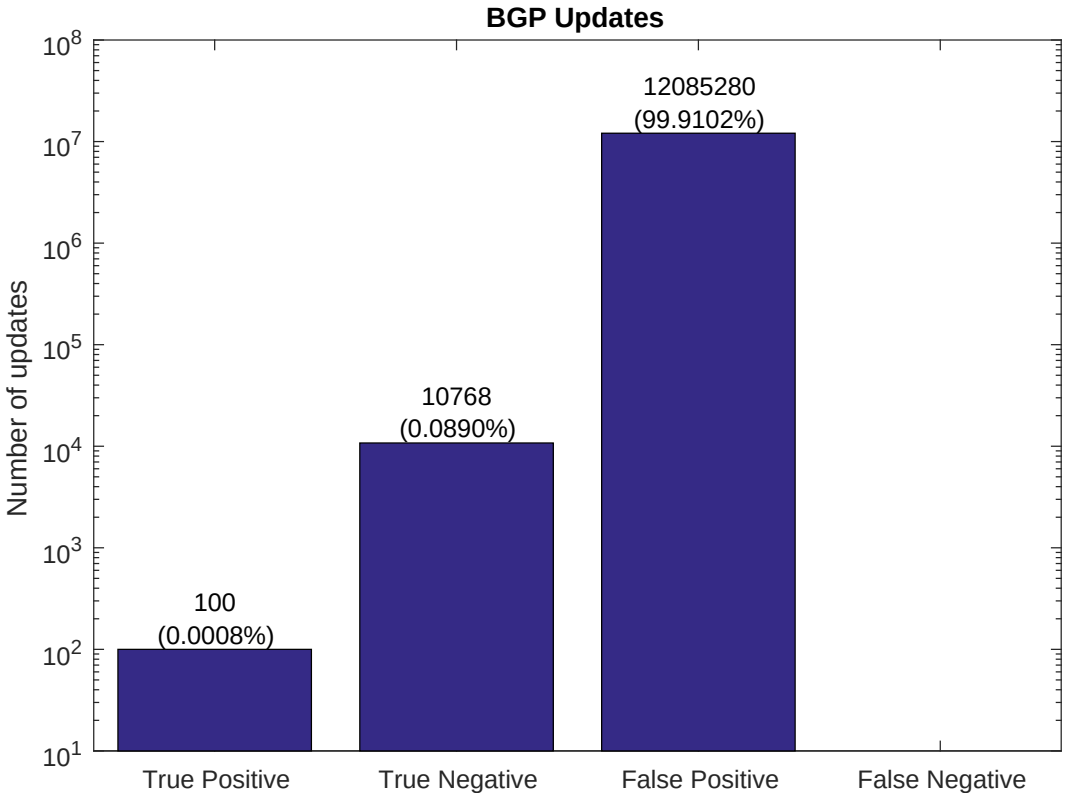


Figure 5.2: BGP Updates - Dataset 1 - 2 Trusted ASes

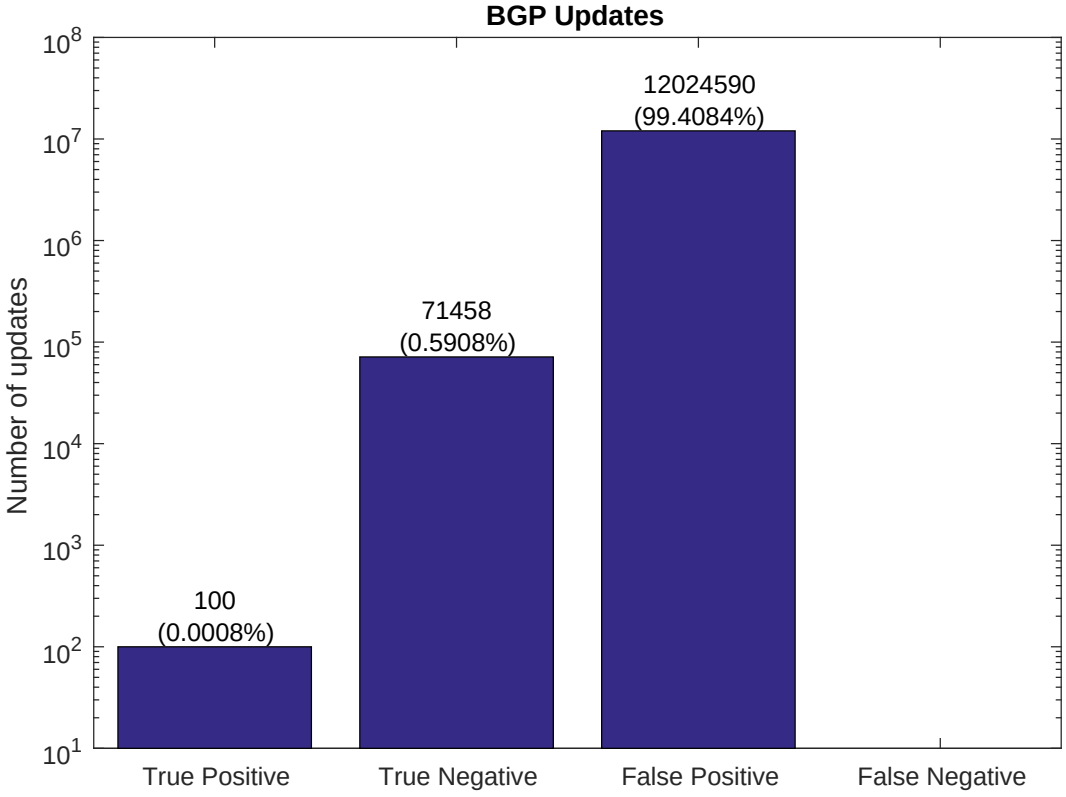


Figure 5.3: BGP Updates - Dataset 1 - 3 Trusted ASes

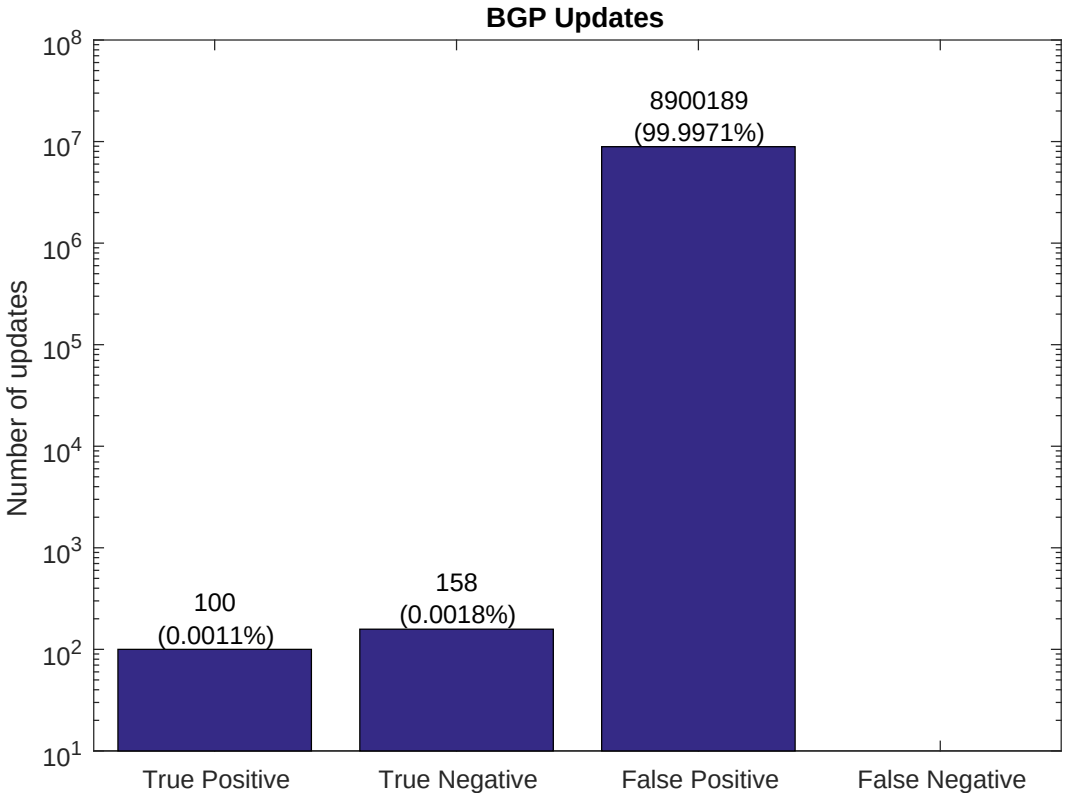


Figure 5.4: BGP Updates - Dataset 2 - 1 Trusted AS

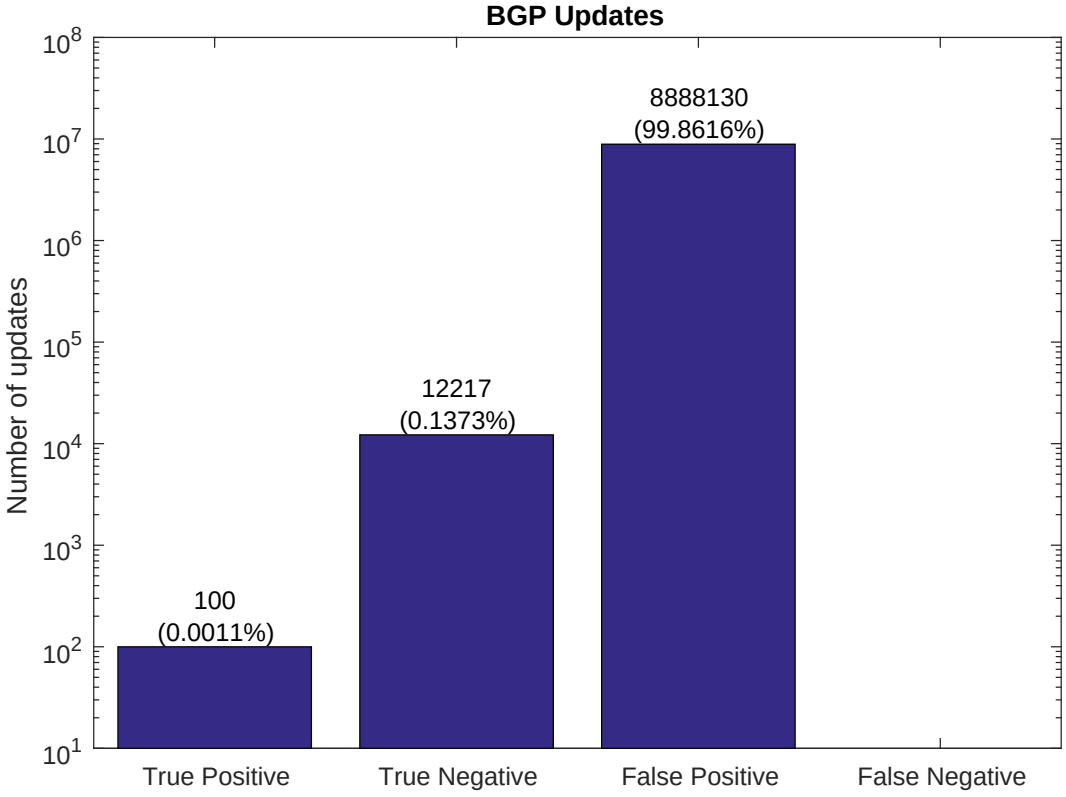


Figure 5.5: BGP Updates - Dataset 2 - 2 Trusted ASes

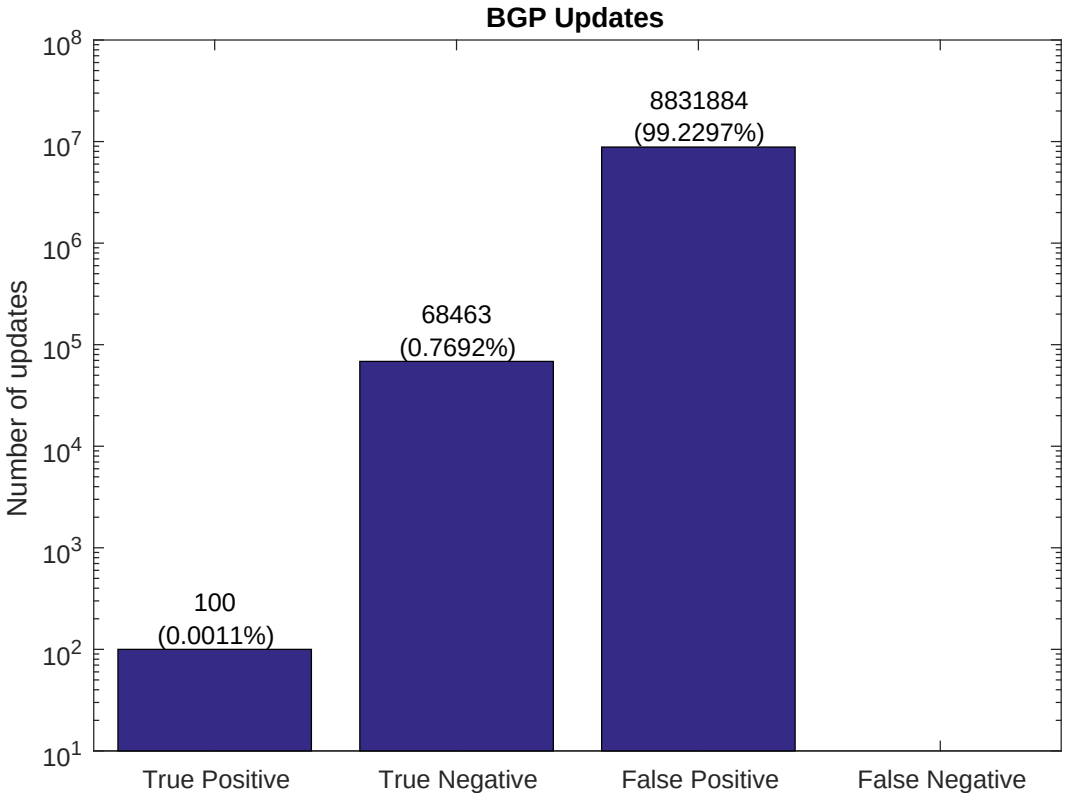


Figure 5.6: BGP Updates - Dataset 2 - 3 Trusted ASes

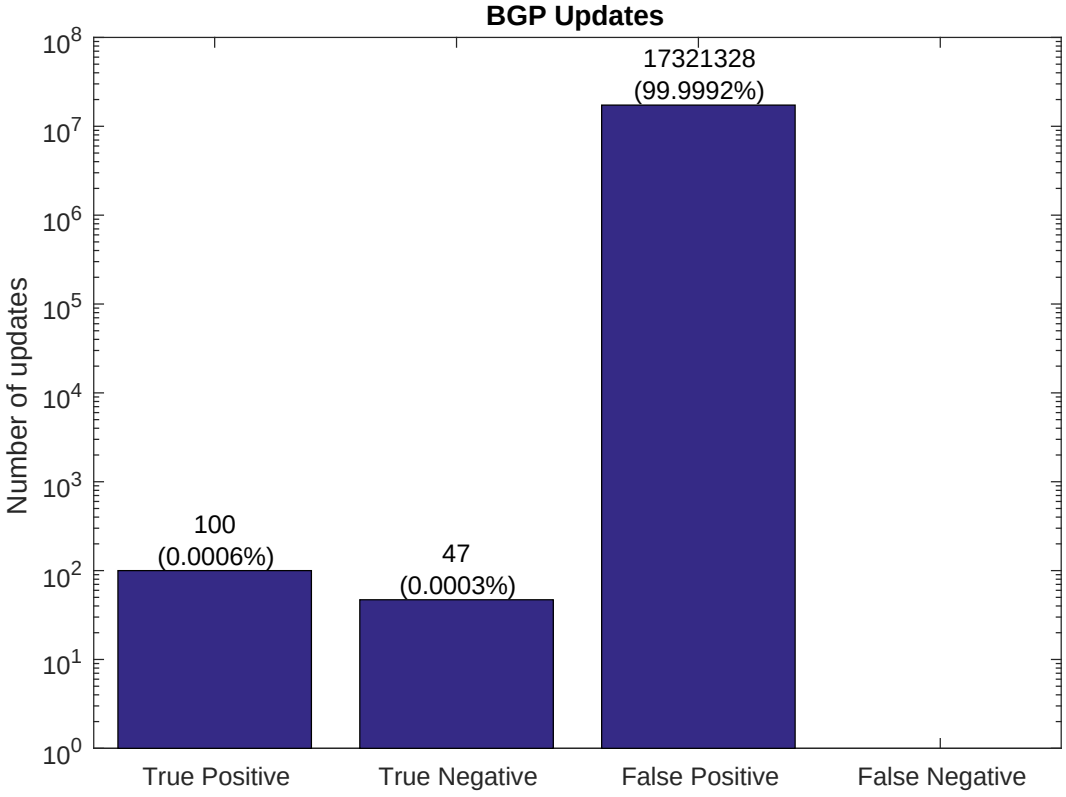


Figure 5.7: BGP Updates - Dataset 3 - 1 Trusted AS

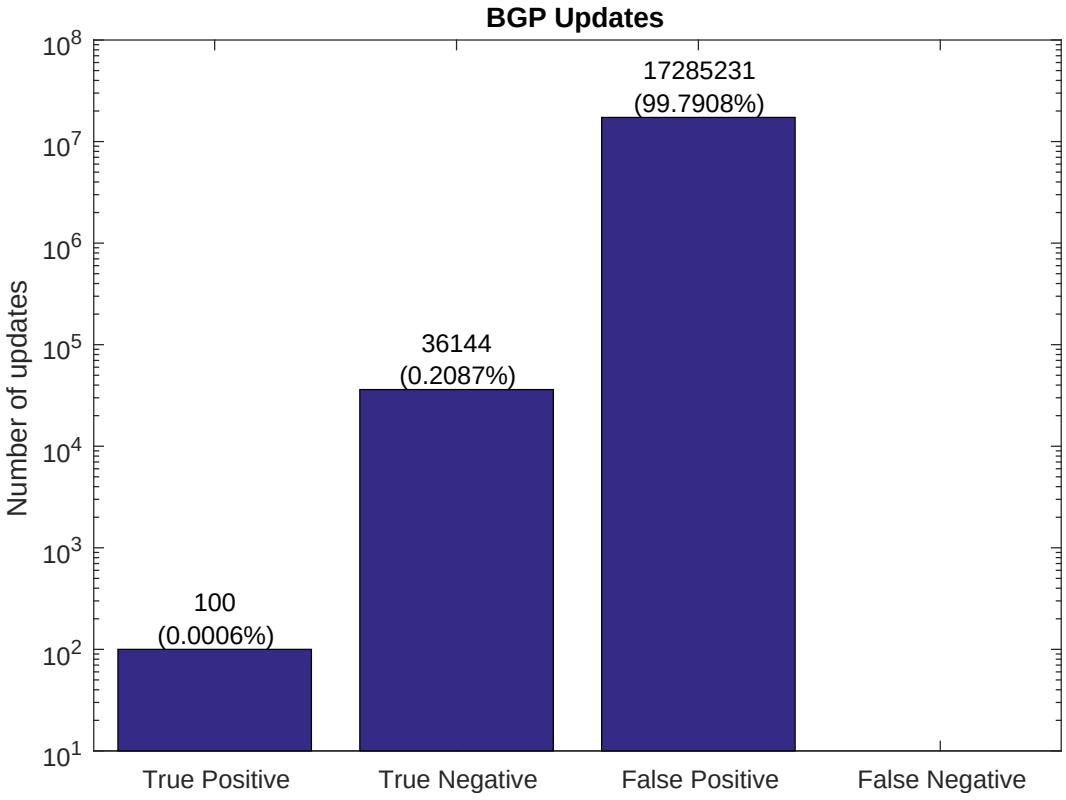


Figure 5.8: BGP Updates - Dataset 3 - 2 Trusted ASes

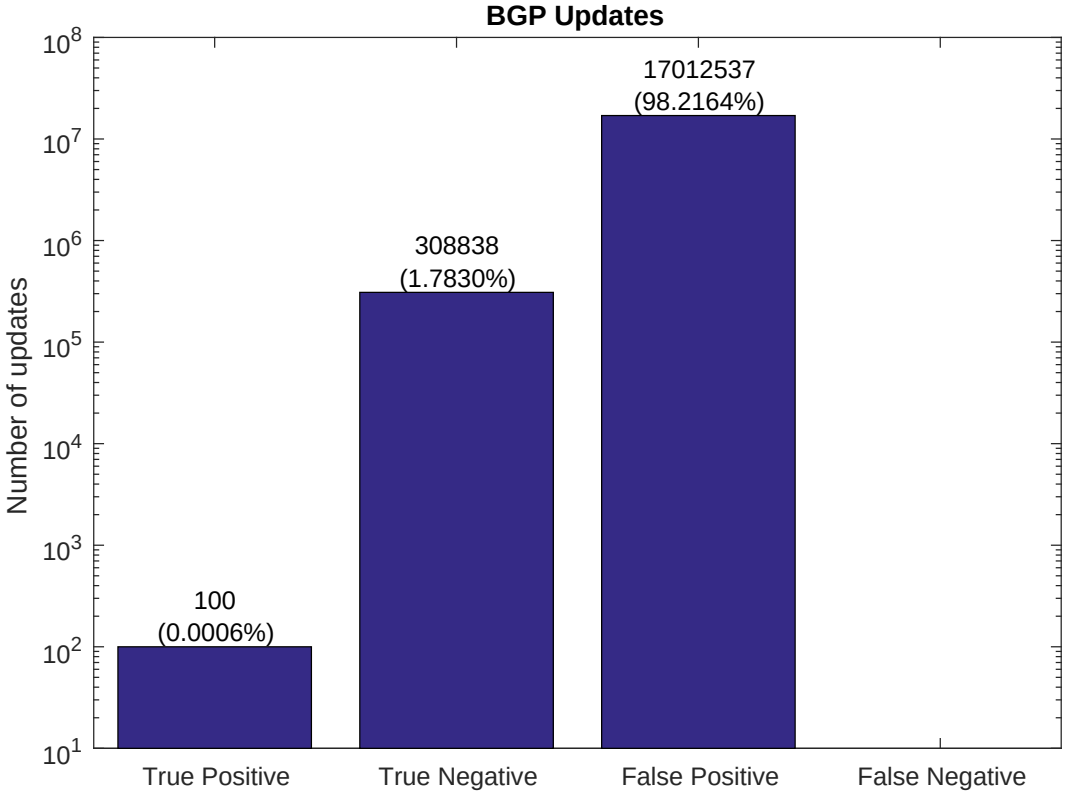


Figure 5.9: BGP Updates - Dataset 3 - 3 Trusted ASes

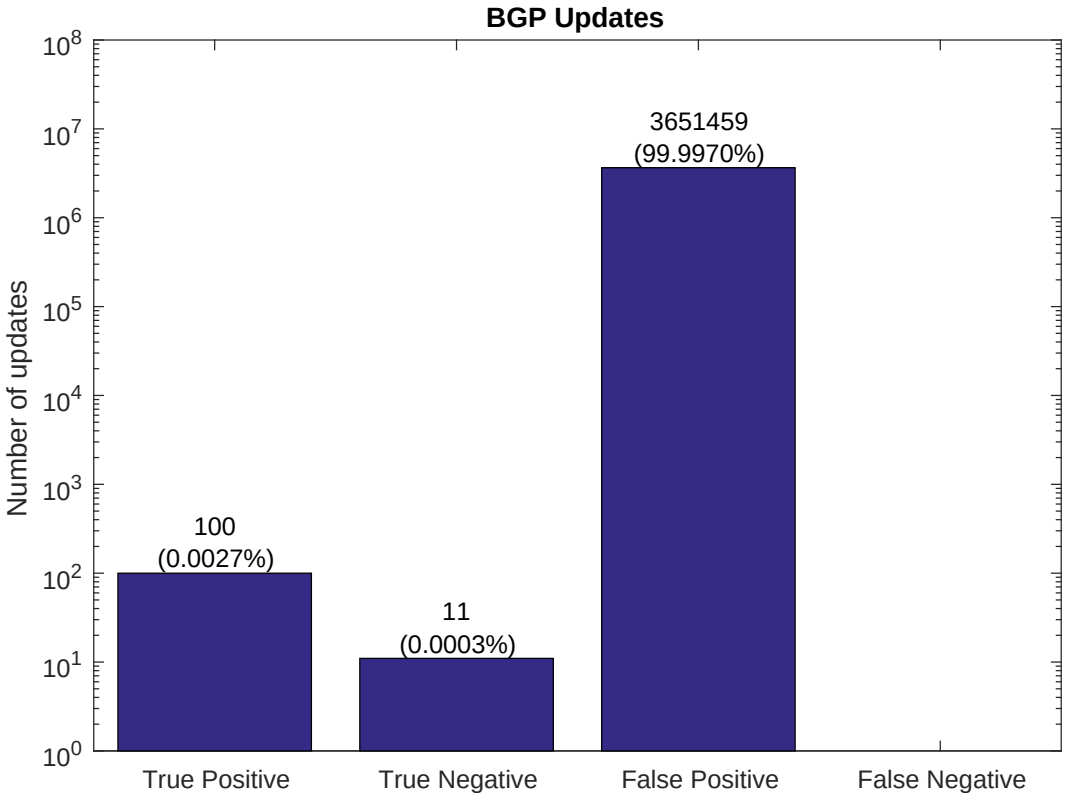


Figure 5.10: BGP Updates - Dataset 4 - 1 Trusted AS

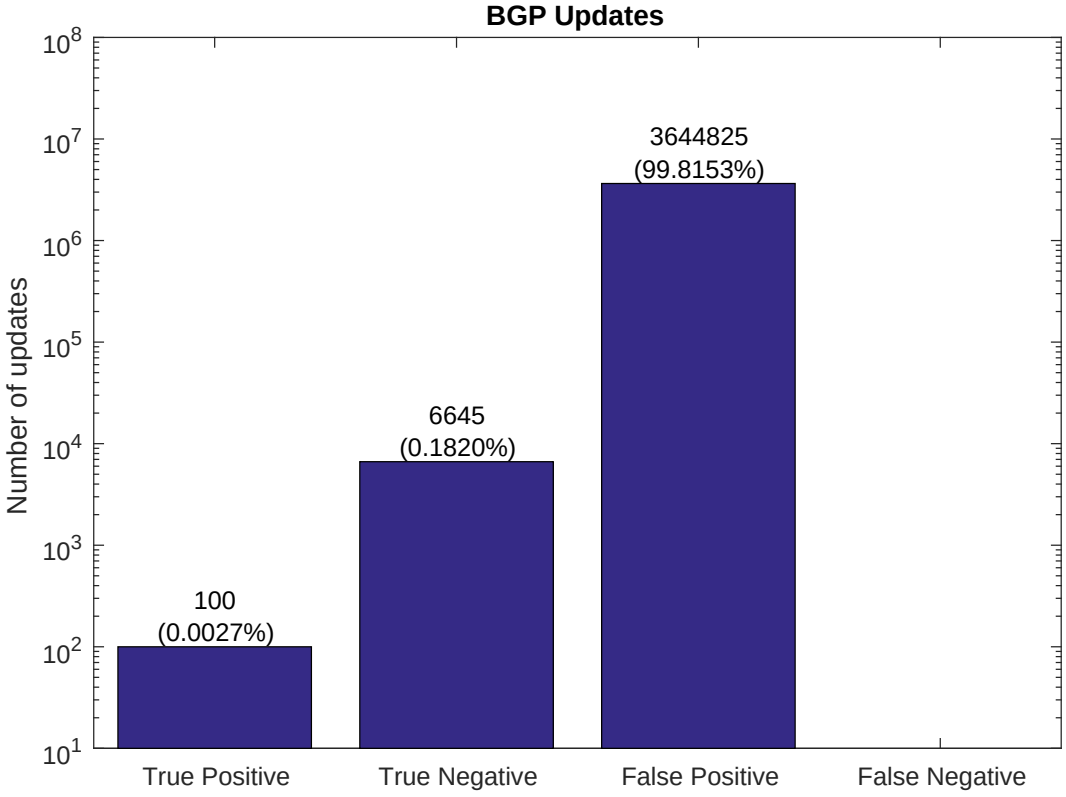


Figure 5.11: BGP Updates - Dataset 4 - 2 Trusted ASes

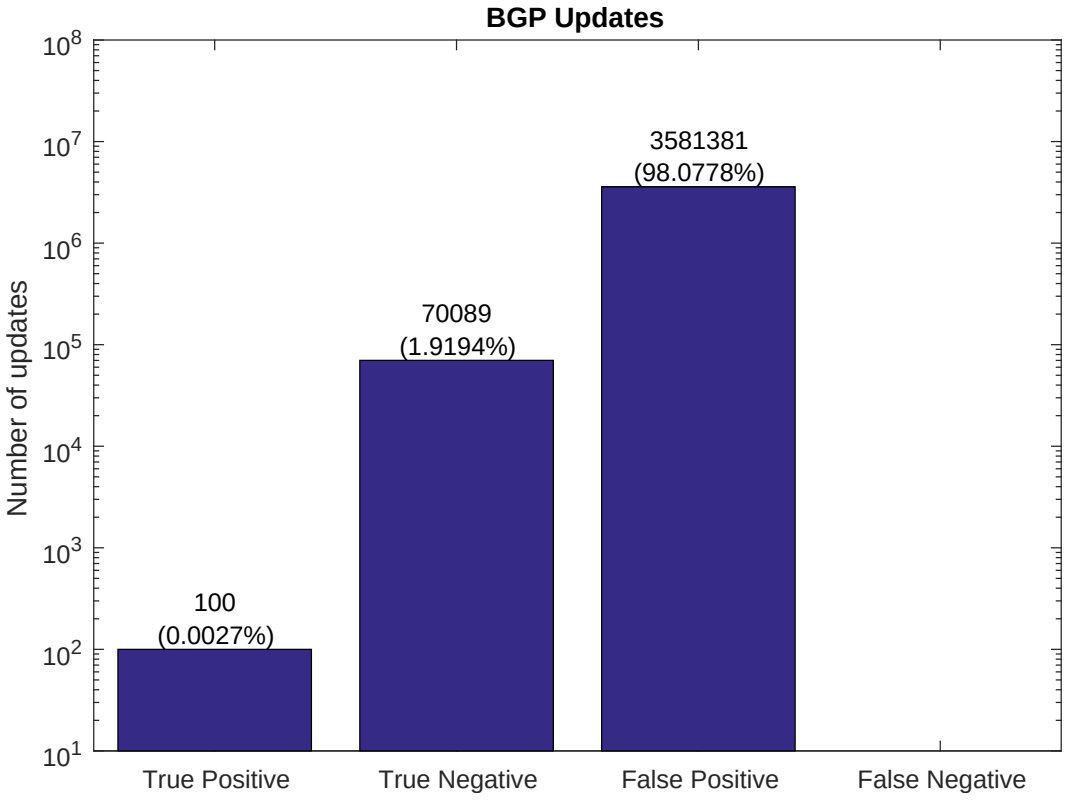


Figure 5.12: BGP Updates - Dataset 4 - 3 Trusted ASes

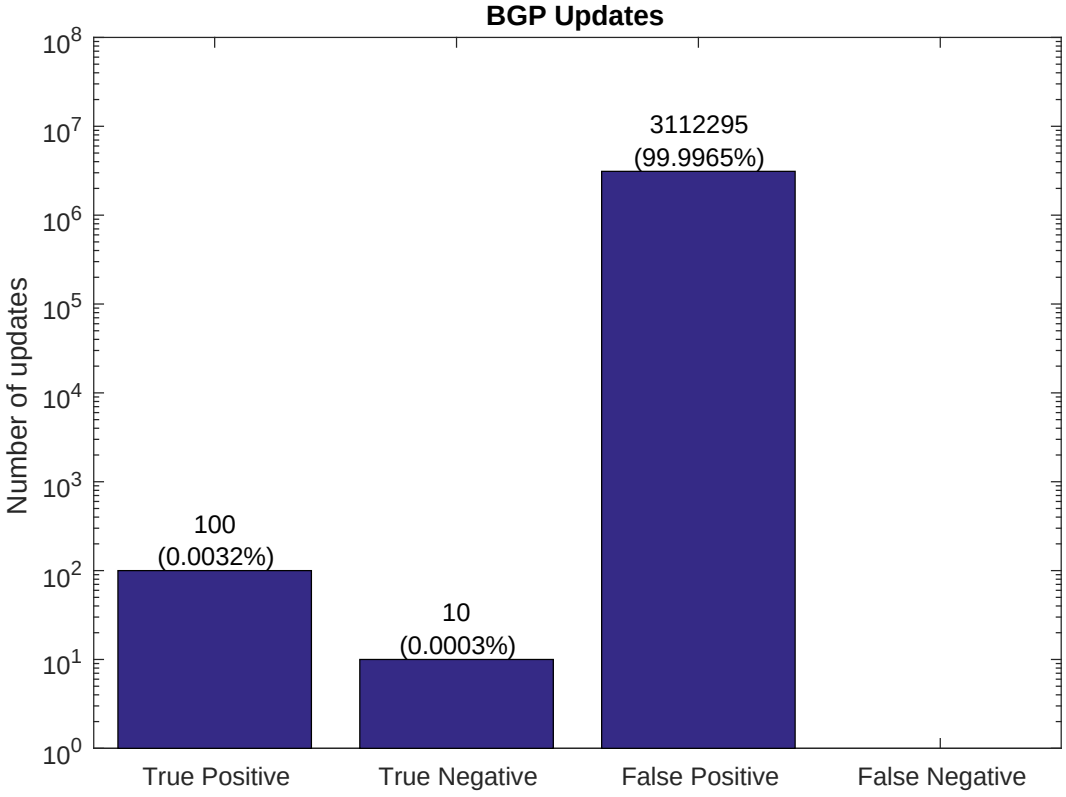


Figure 5.13: BGP Updates - Dataset 5 - 1 Trusted AS

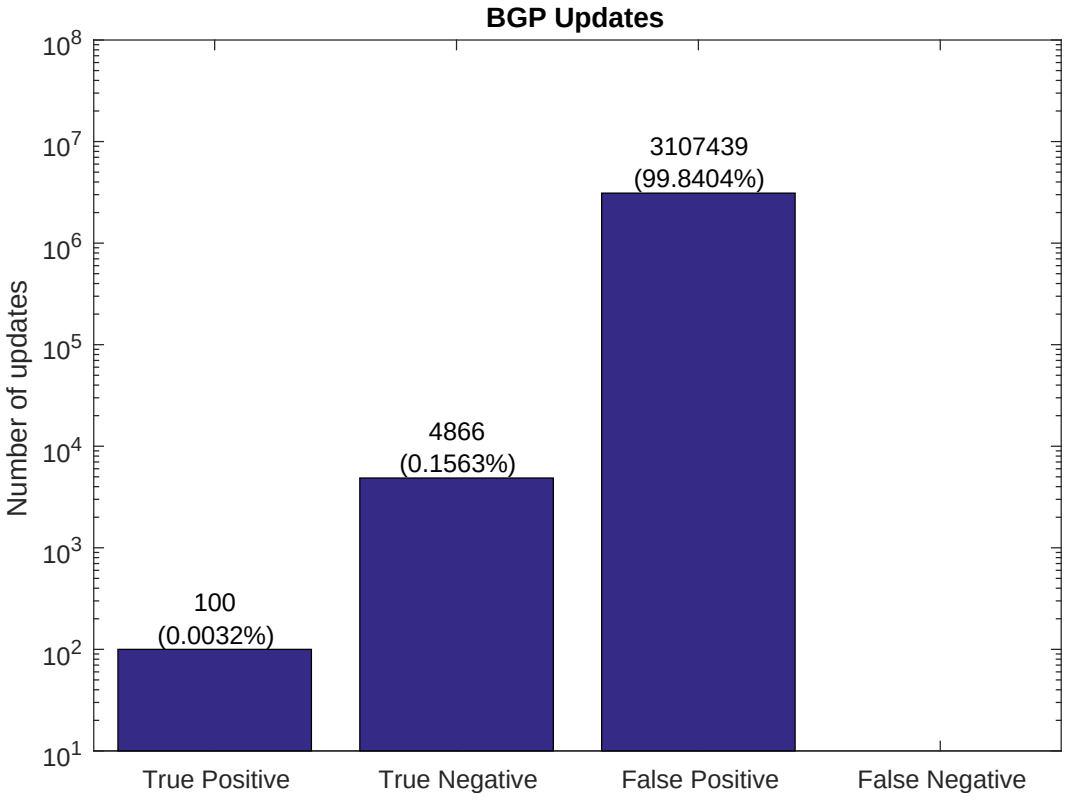


Figure 5.14: BGP Updates - Dataset 5 - 2 Trusted ASes

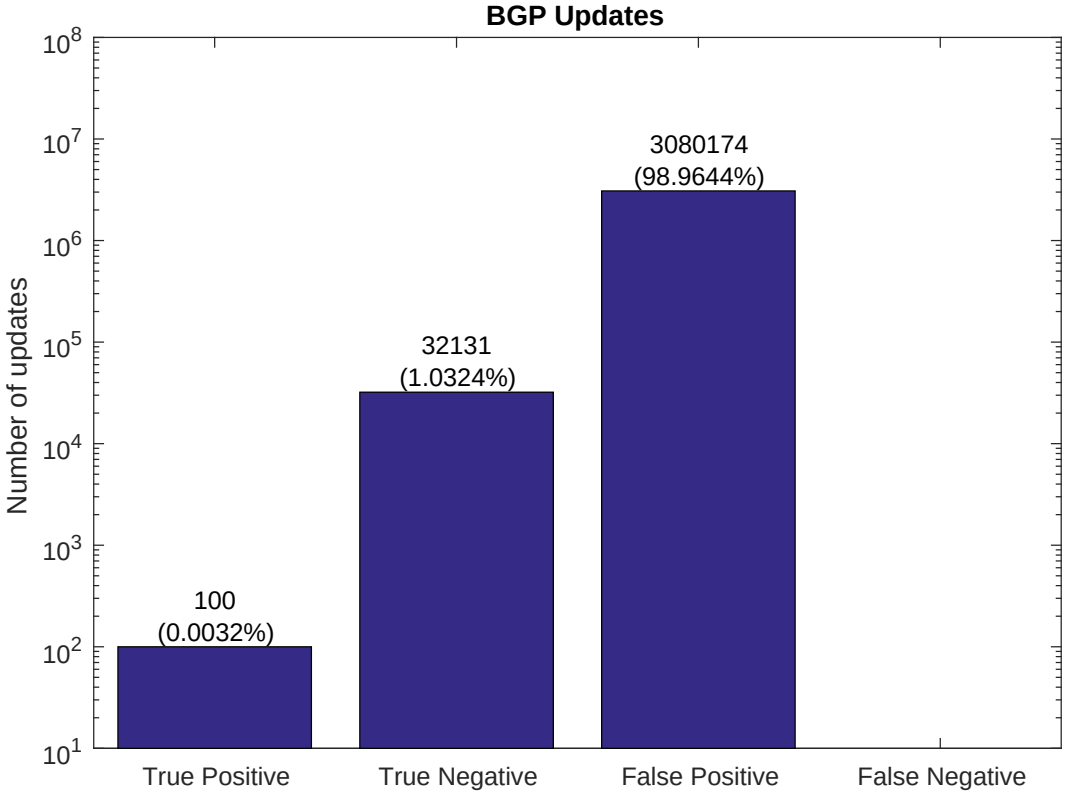


Figure 5.15: BGP Updates - Dataset 5 - 3 Trusted ASes

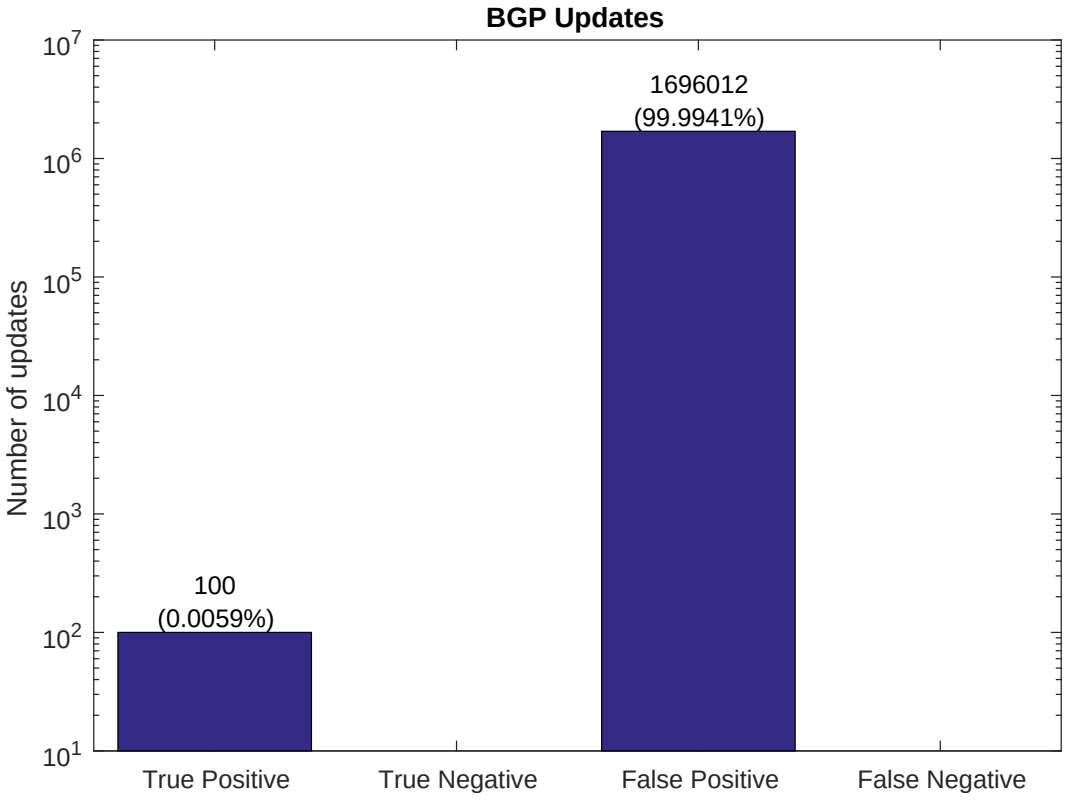


Figure 5.16: BGP Updates - Dataset 6 - 1 Trusted AS

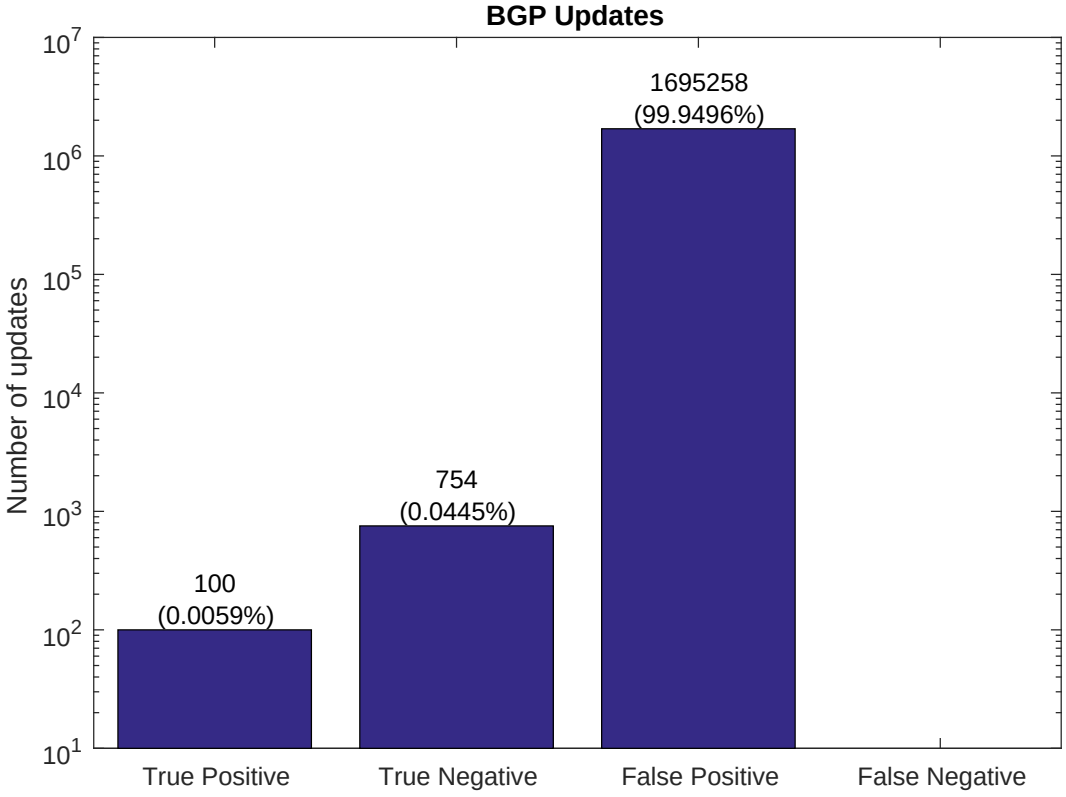


Figure 5.17: BGP Updates - Dataset 6 - 2 Trusted ASes

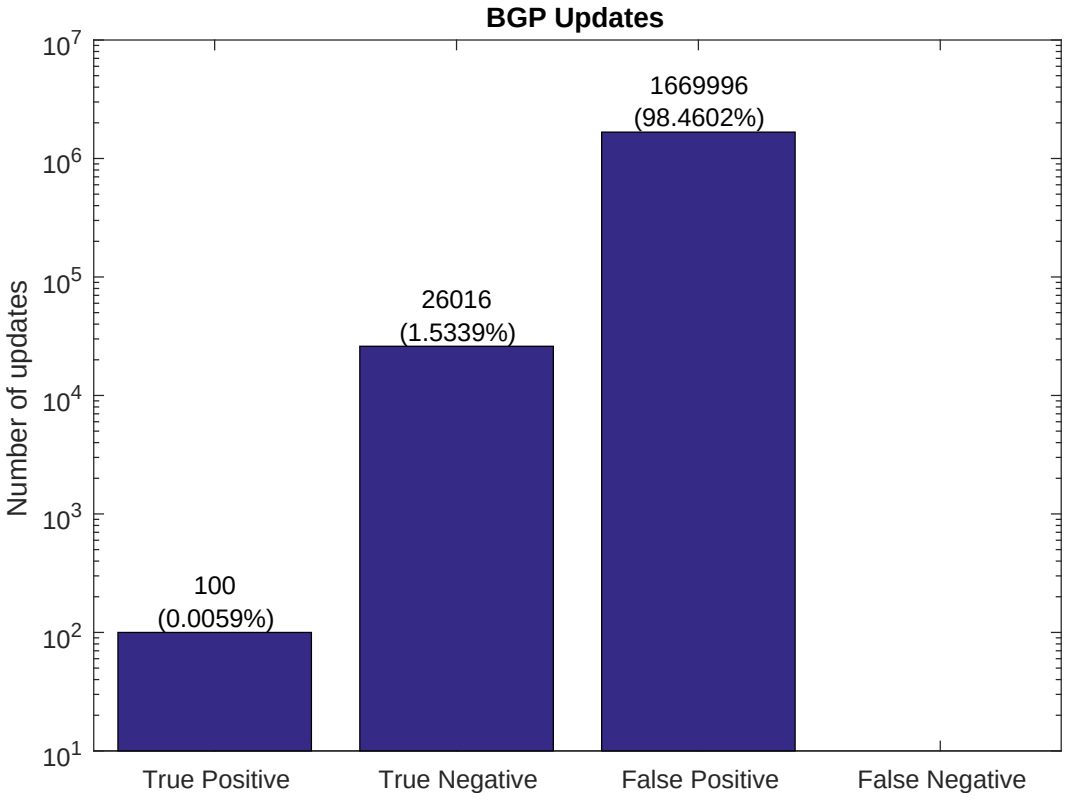


Figure 5.18: BGP Updates - Dataset 6 - 3 Trusted ASes

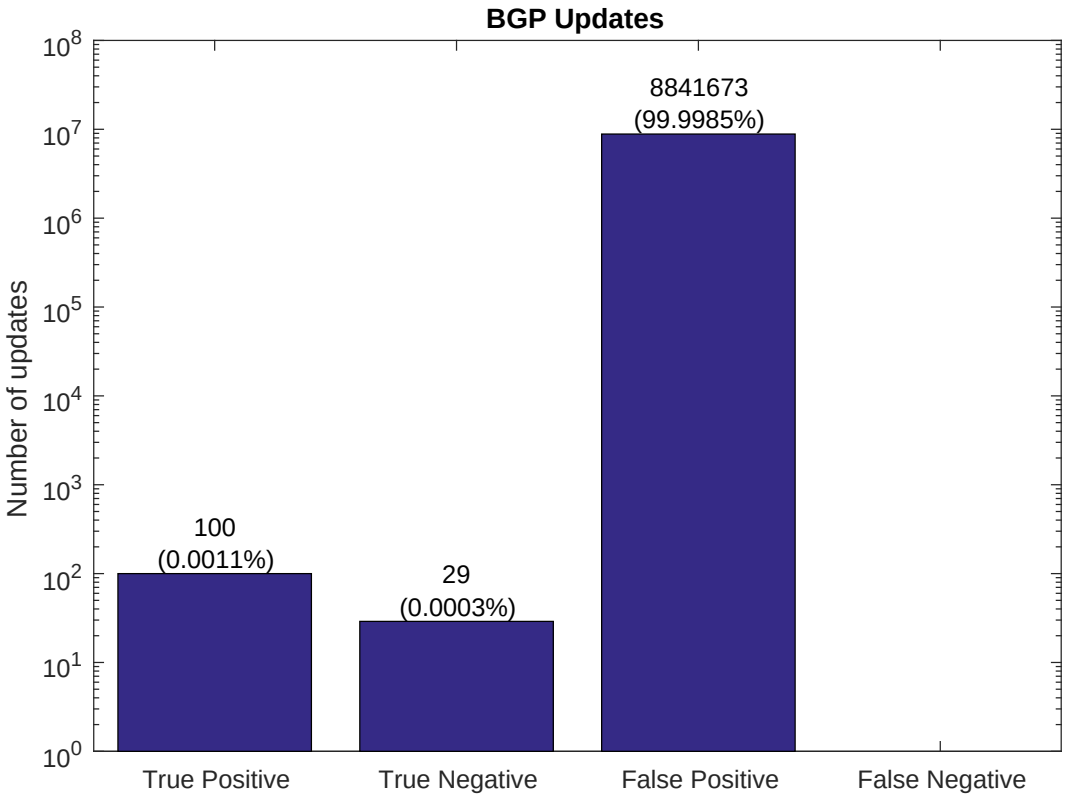


Figure 5.19: BGP Updates - Dataset 7 - 1 Trusted AS

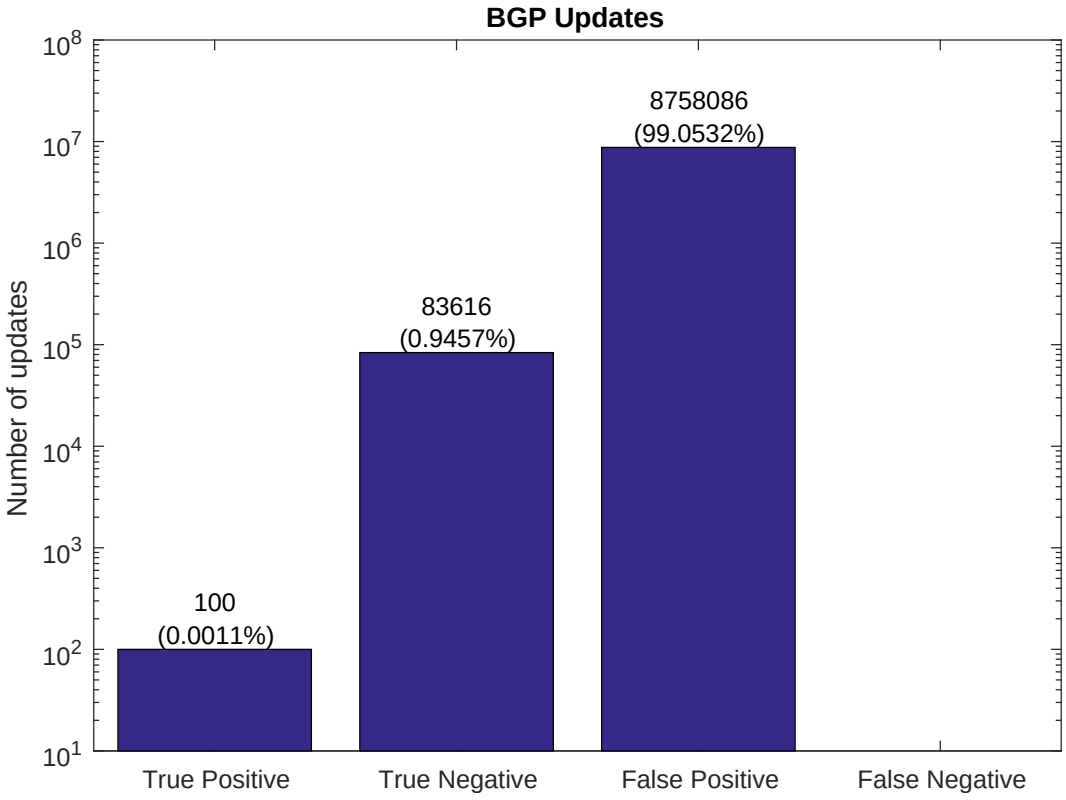


Figure 5.20: BGP Updates - Dataset 7 - 2 Trusted ASes

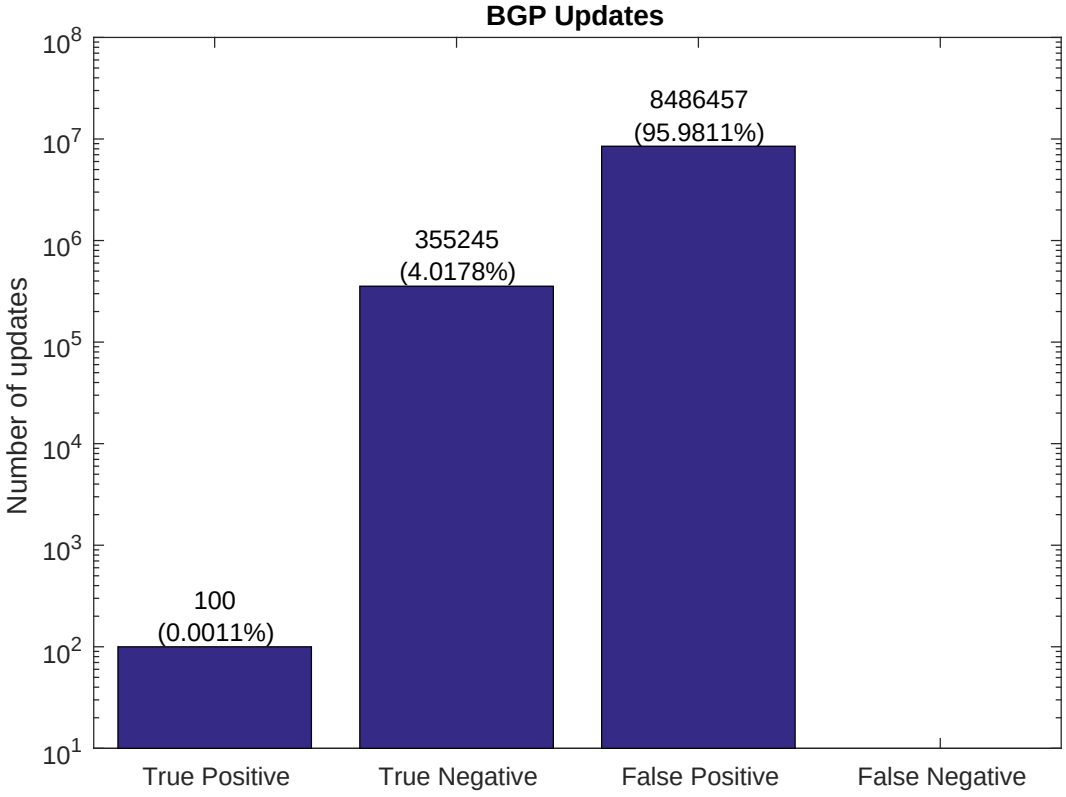


Figure 5.21: BGP Updates - Dataset 7 - 3 Trusted ASes

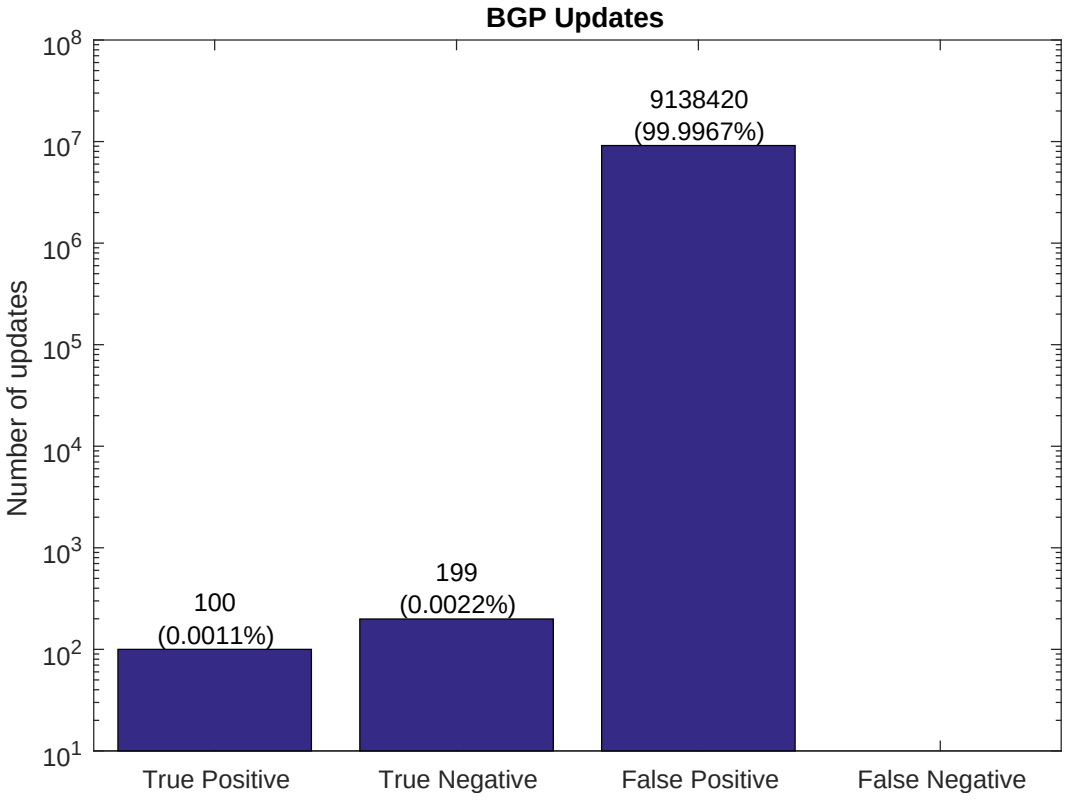


Figure 5.22: BGP Updates - Dataset 8 - 1 Trusted AS

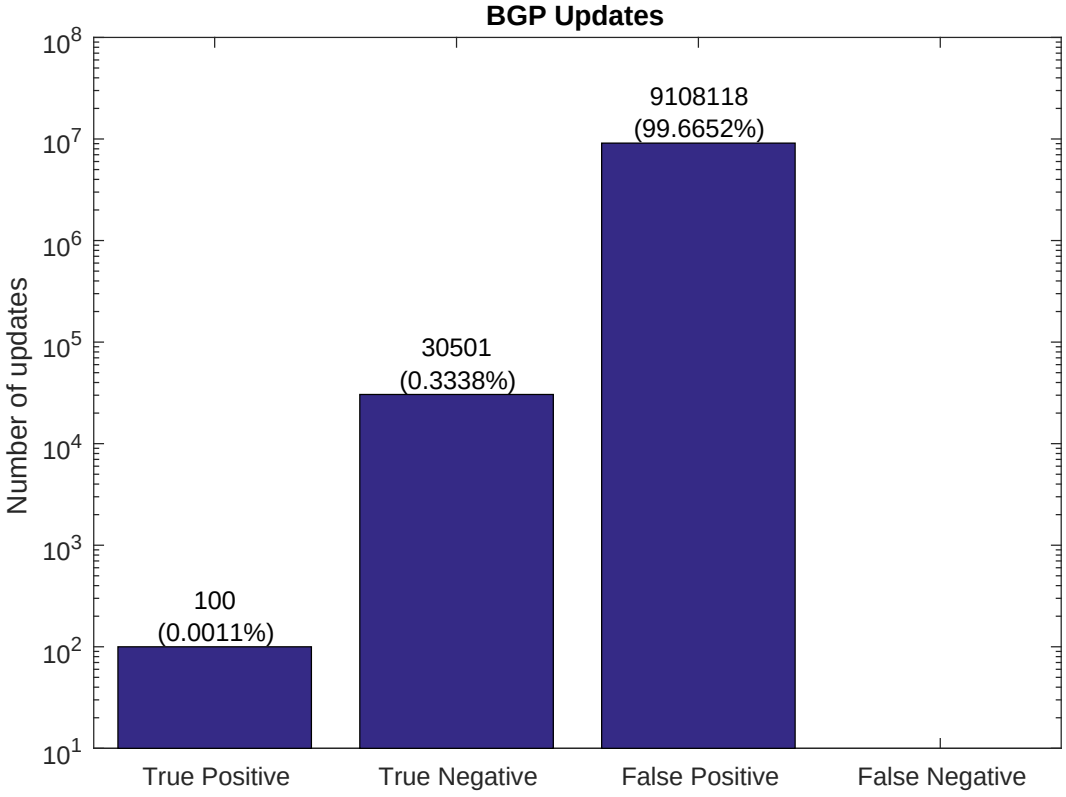


Figure 5.23: BGP Updates - Dataset 8 - 2 Trusted ASes

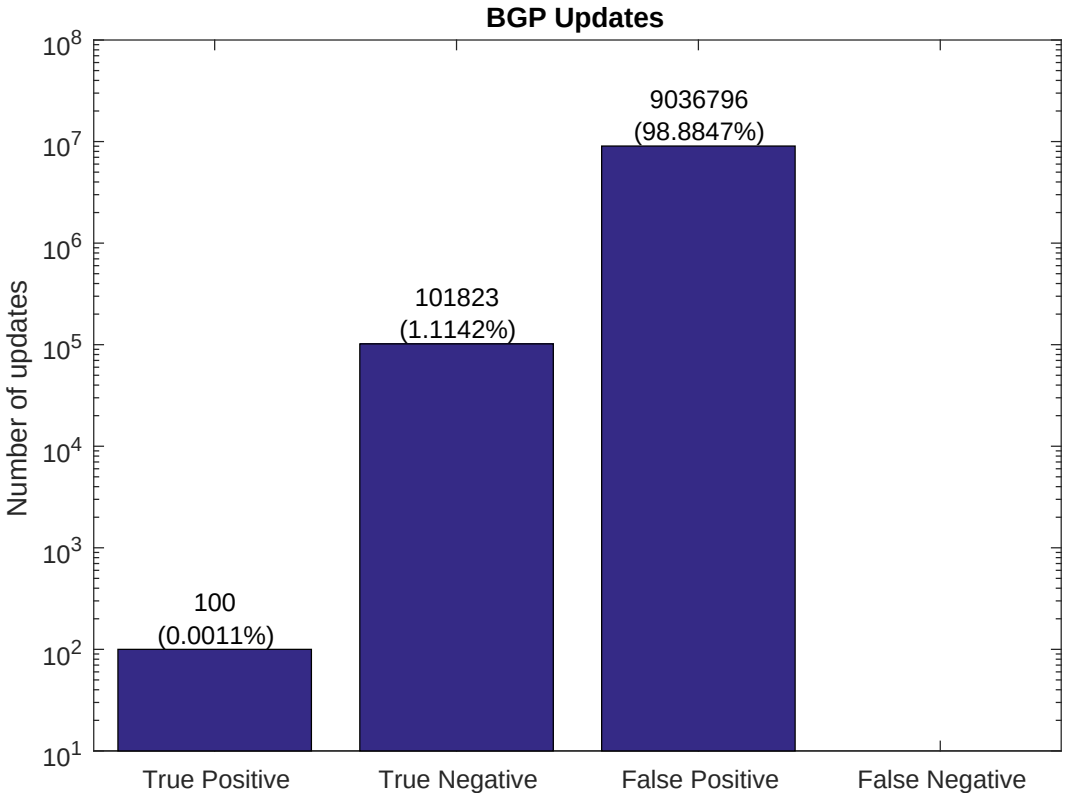


Figure 5.24: BGP Updates - Dataset 8 - 3 Trusted ASes

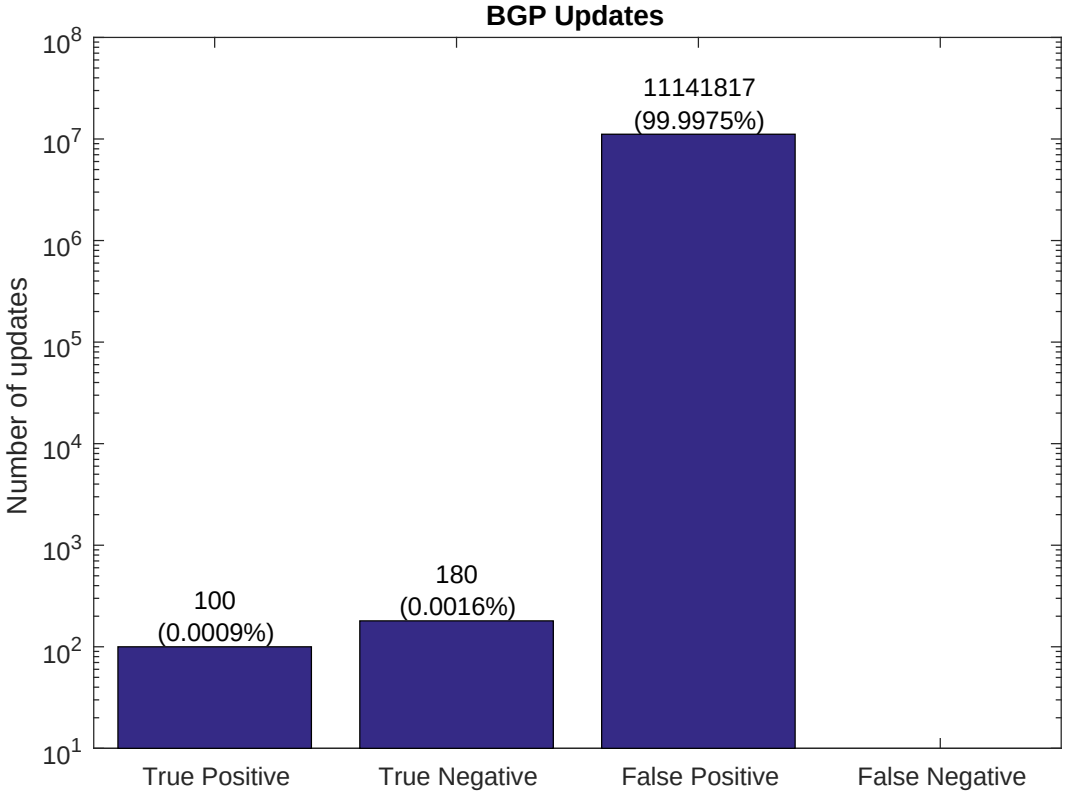


Figure 5.25: BGP Updates - Dataset 9 - 1 Trusted AS

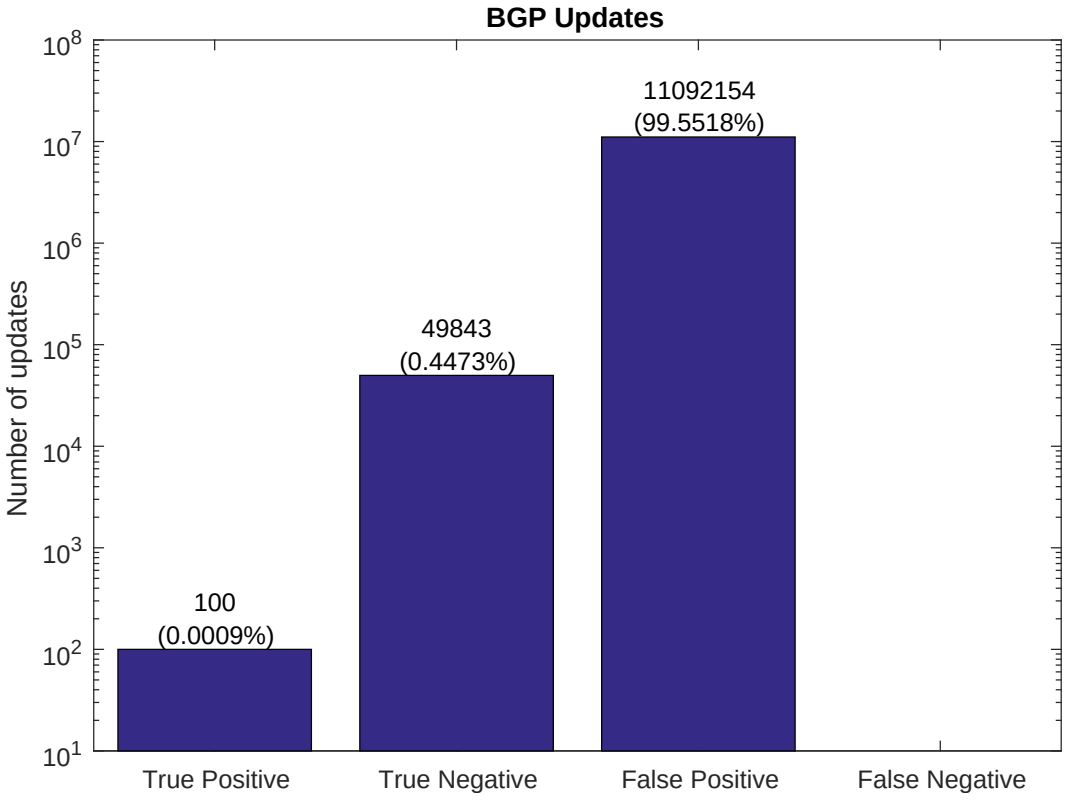


Figure 5.26: BGP Updates - Dataset 9 - 2 Trusted ASes

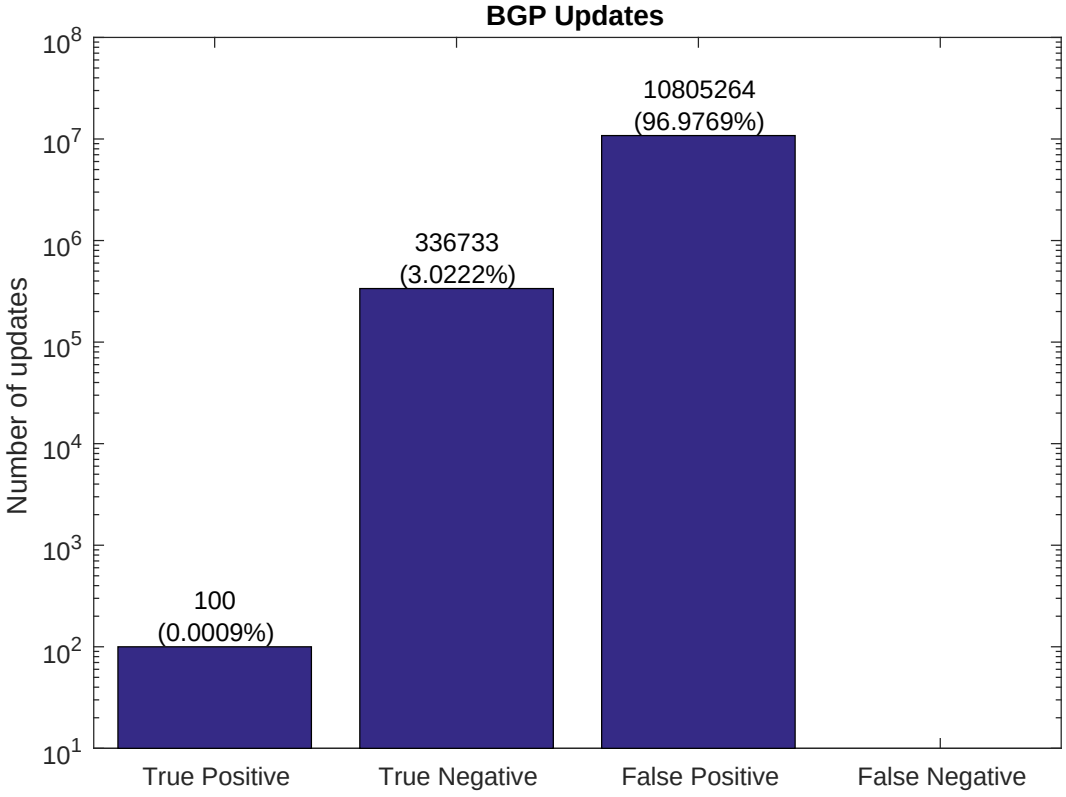


Figure 5.27: BGP Updates - Dataset 9 - 3 Trusted ASes

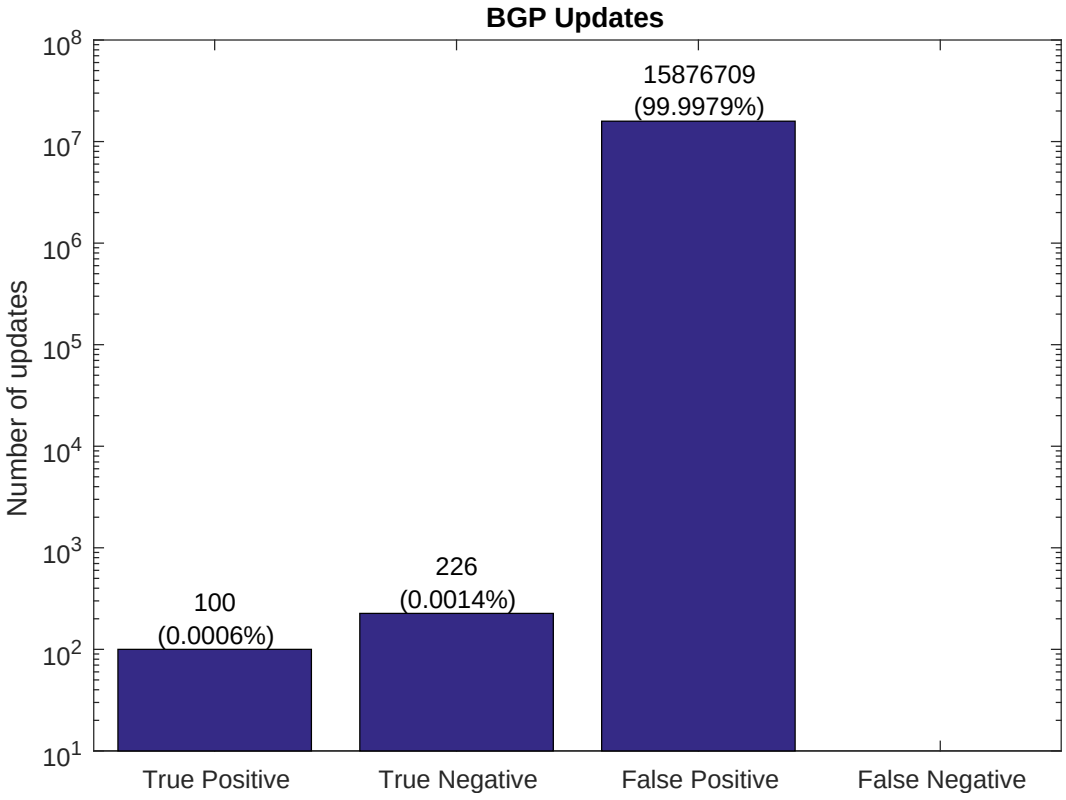


Figure 5.28: BGP Updates - Dataset 10 - 1 Trusted AS

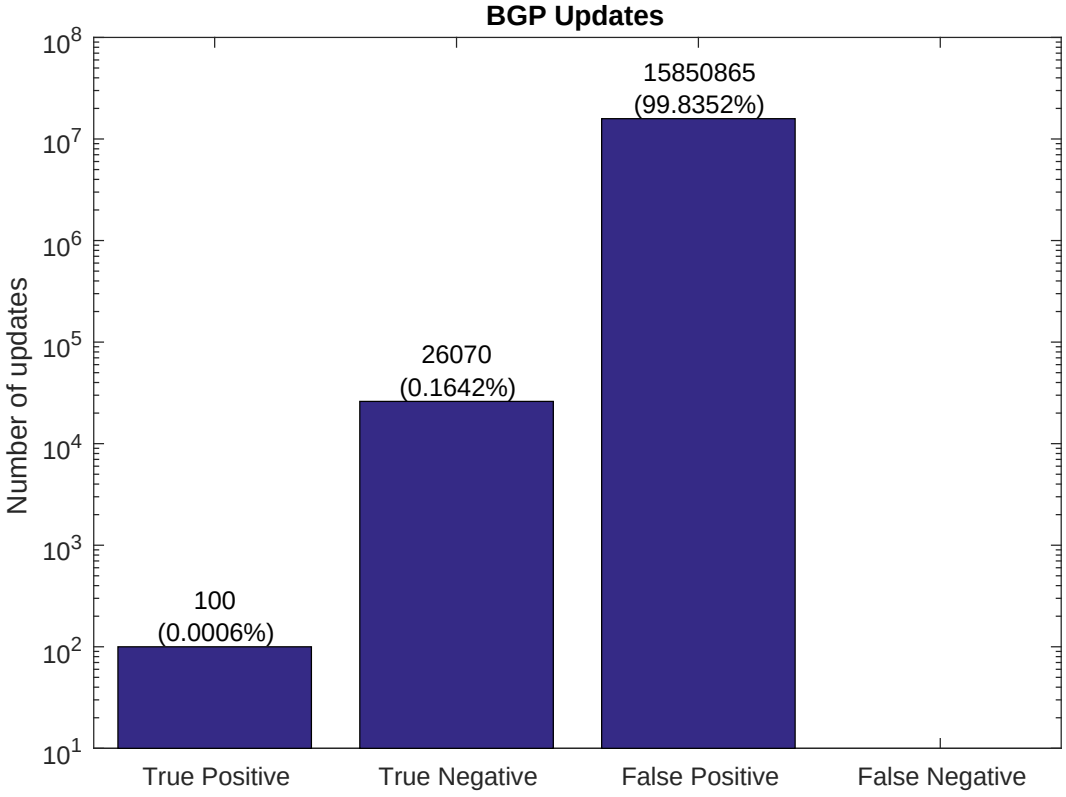


Figure 5.29: BGP Updates - Dataset 10 - 2 Trusted ASes

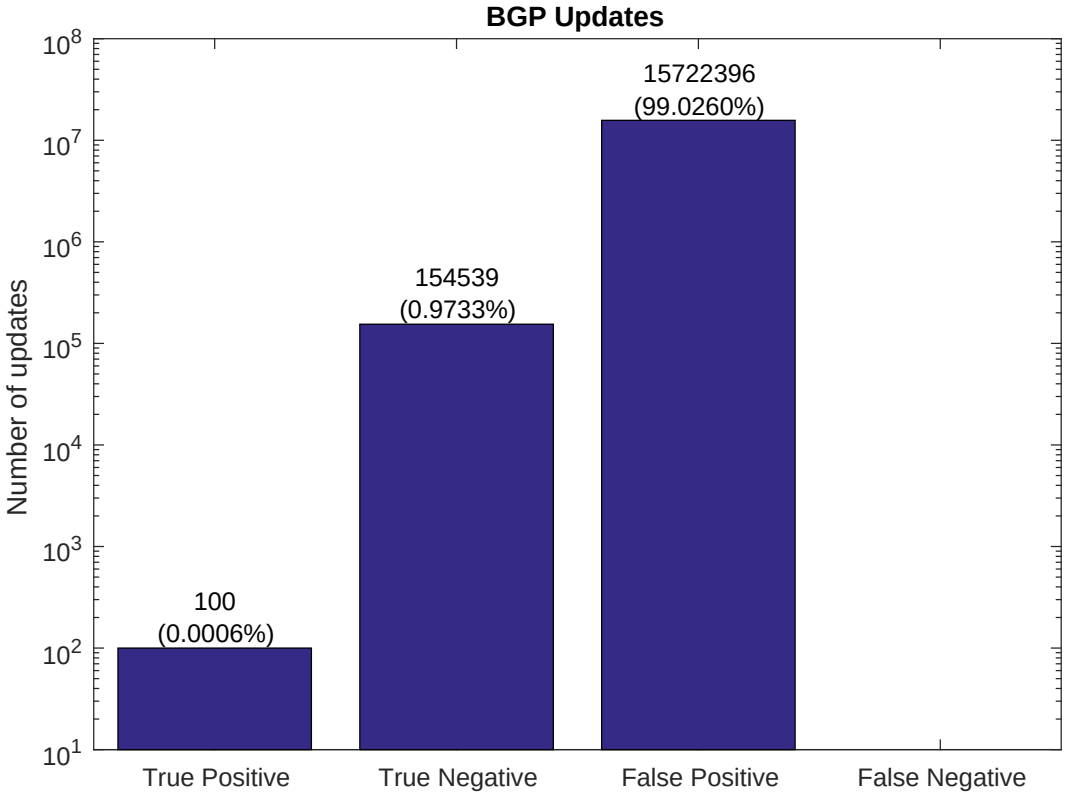


Figure 5.30: BGP Updates - Dataset 10 - 3 Trusted ASes

Data on Partially Verified BGP Updates

The subsequent graphs present the raw data regarding the BGP Updates that are partially verified by others already stored on the Blockchain. This data is in percentages. The blue bars represent the percentages in relation to the amount of updates that were verified. The yellow bars represent the percentages in relation to the total updates in the dataset utilized.

Each graph corresponds to a different dataset, which was executed three times, when considering 1, 2, or 3 ASes as initially trusted by the algorithm.

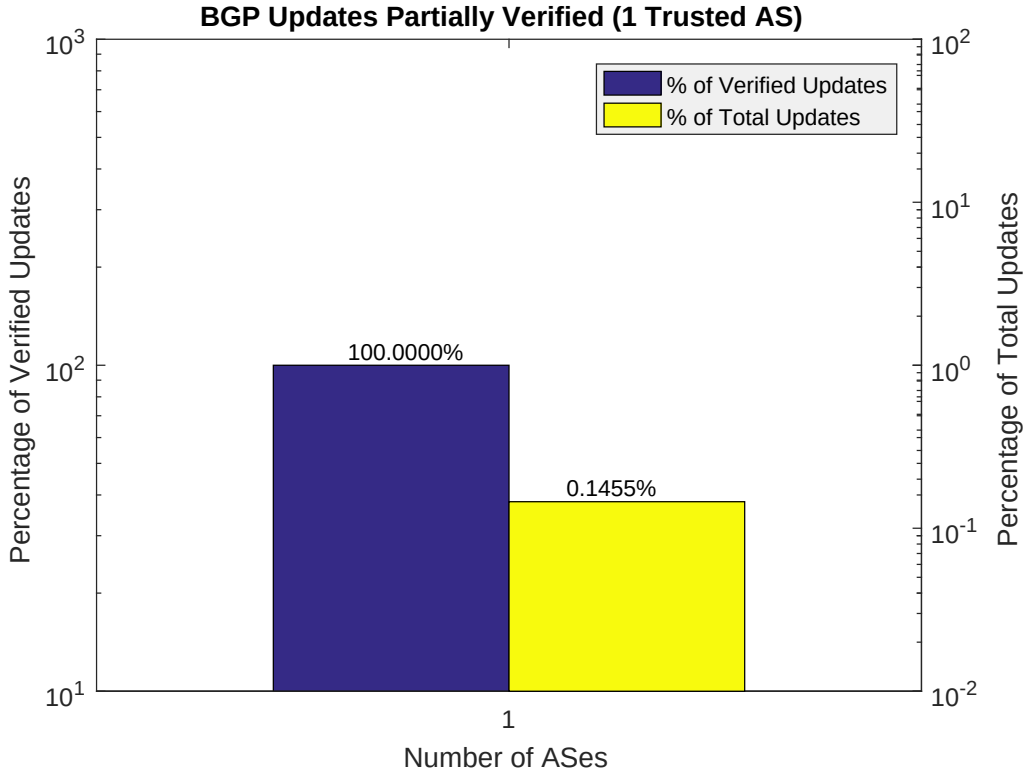


Figure 5.31: Partially Verified BGP Updates - Dataset 1 - 1 Trusted AS

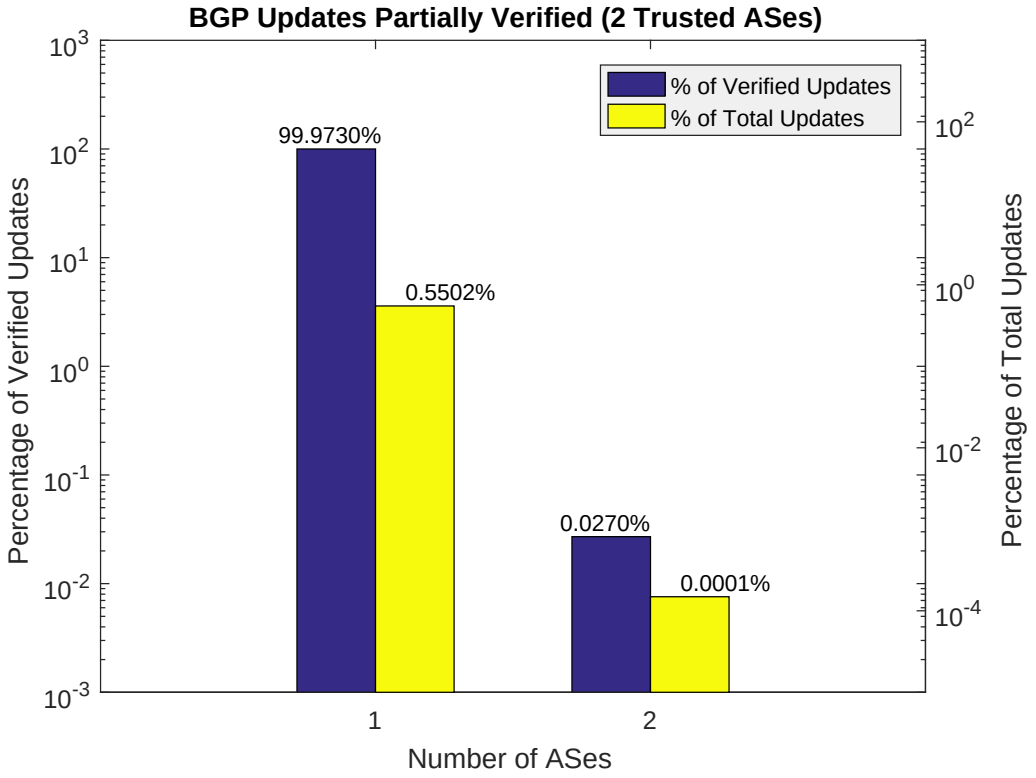


Figure 5.32: Partially Verified BGP Updates - Dataset 1 - 2 Trusted ASes

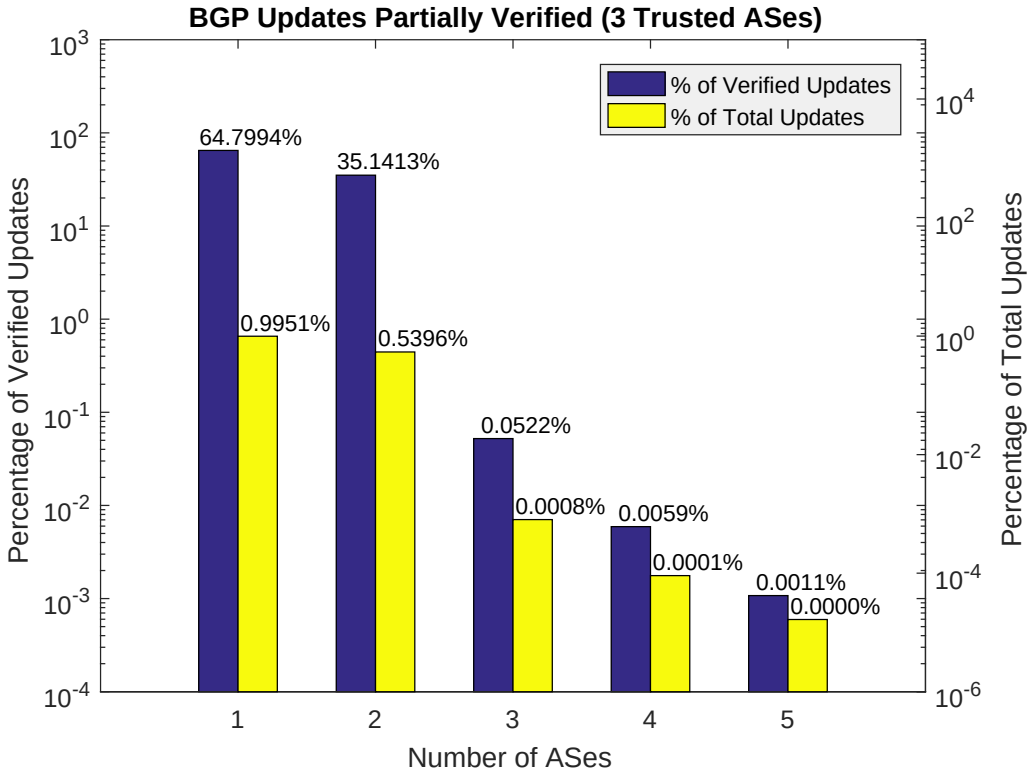


Figure 5.33: Partially Verified BGP Updates - Dataset 1 - 3 Trusted ASes

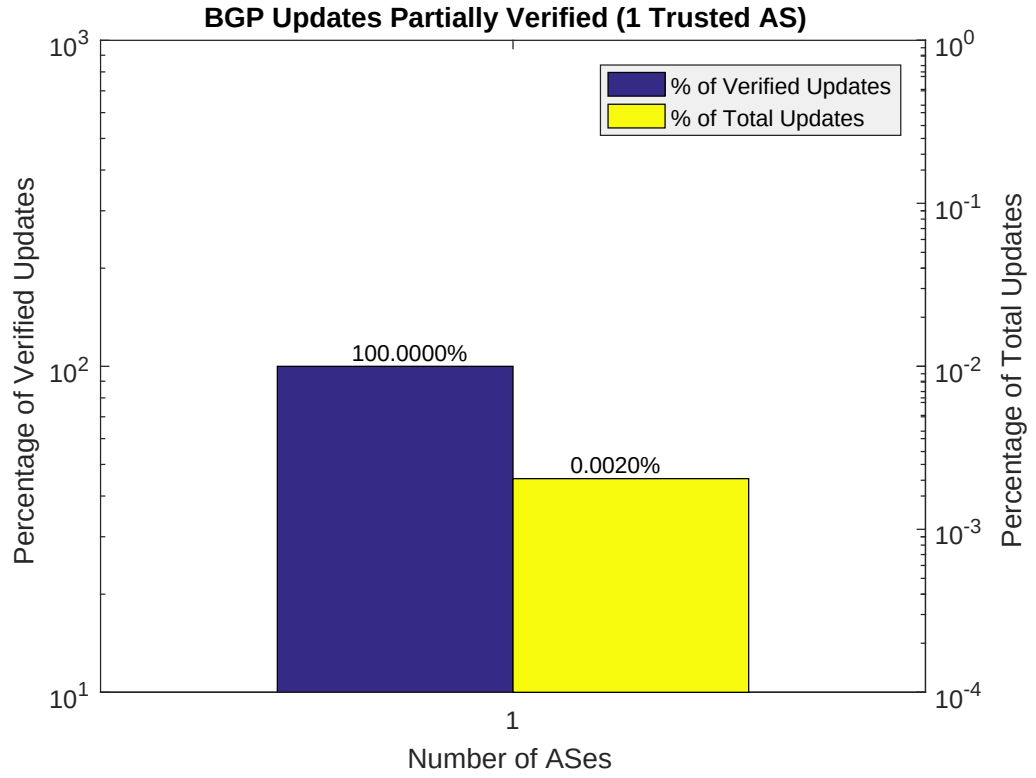


Figure 5.34: Partially Verified BGP Updates - Dataset 2 - 1 Trusted AS

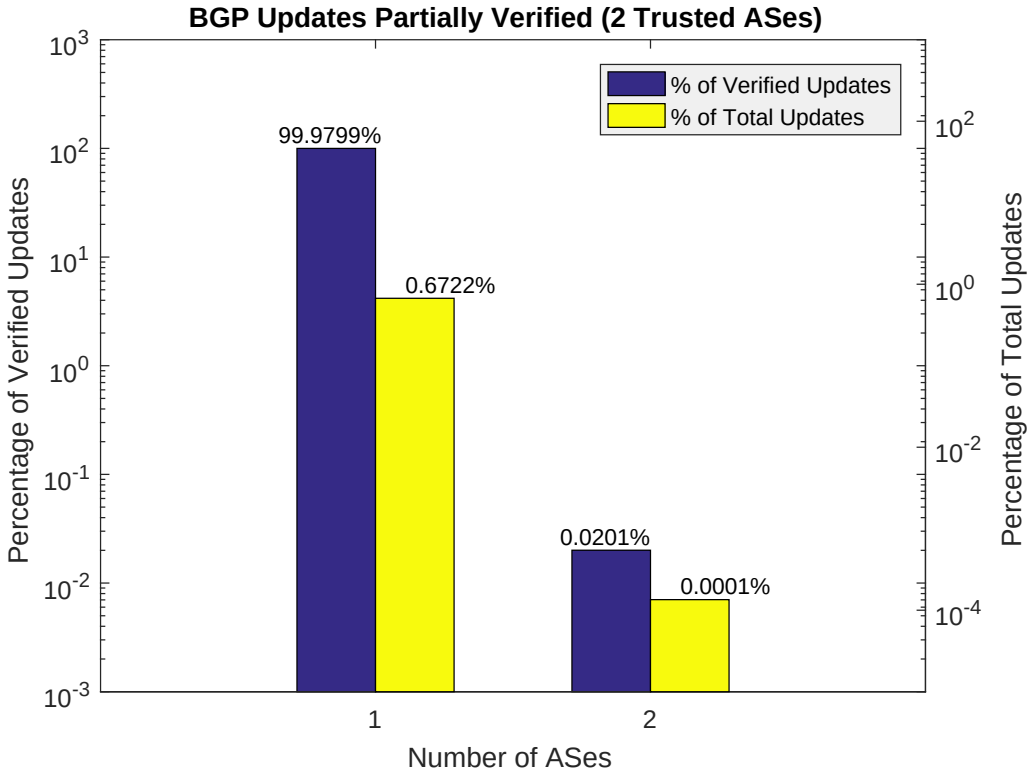


Figure 5.35: Partially Verified BGP Updates - Dataset 2 - 2 Trusted ASes

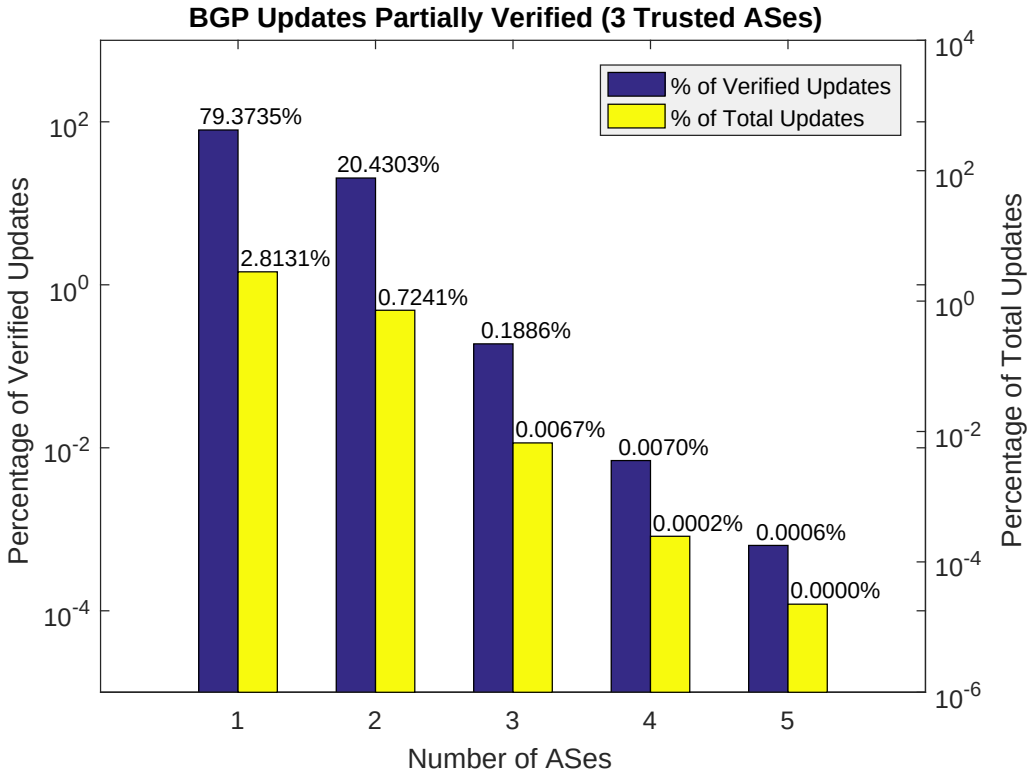


Figure 5.36: Partially Verified BGP Updates - Dataset 2 - 3 Trusted ASes

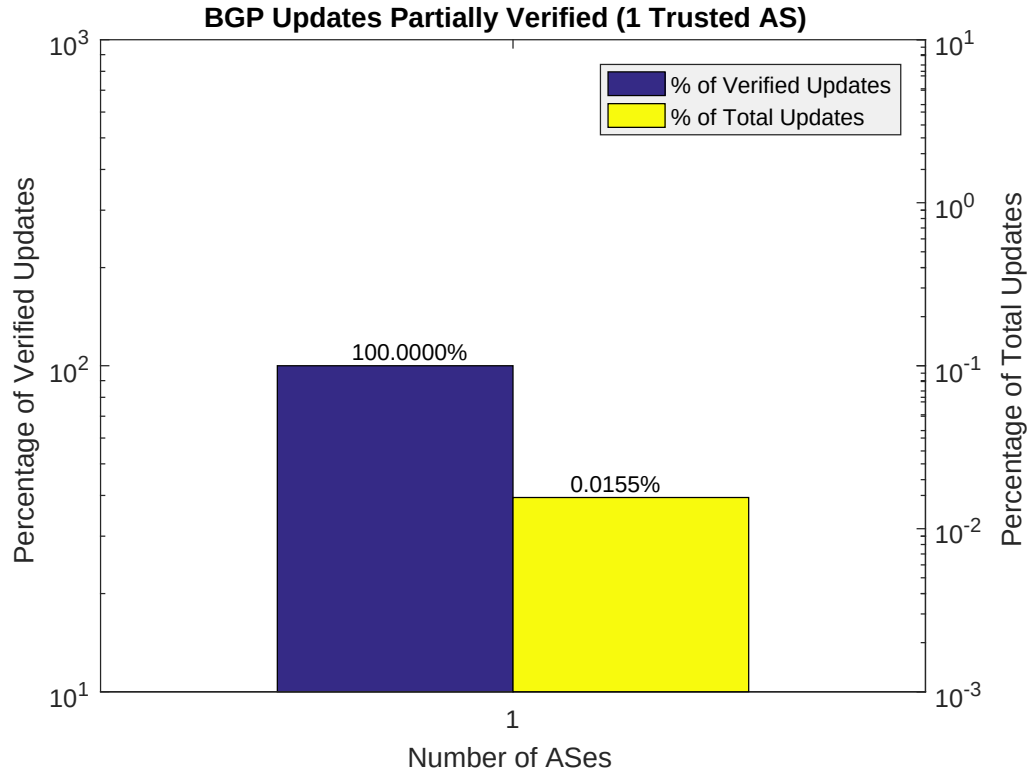


Figure 5.37: Partially Verified BGP Updates - Dataset 3 - 1 Trusted AS

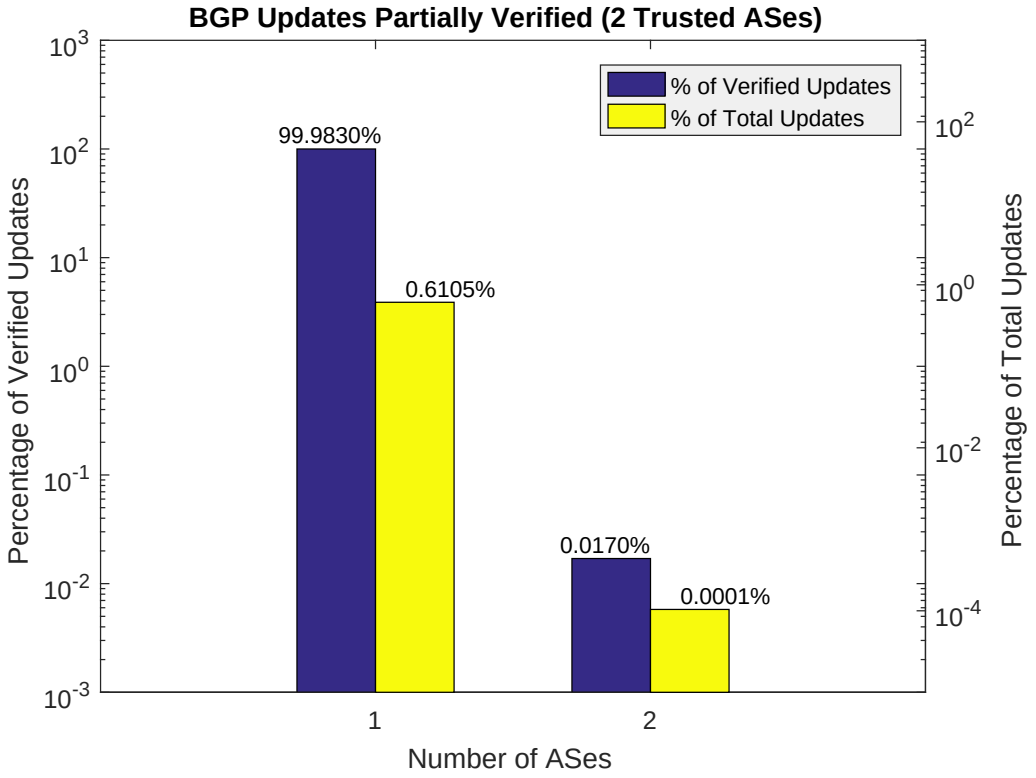


Figure 5.38: Partially Verified BGP Updates - Dataset 3 - 2 Trusted ASes

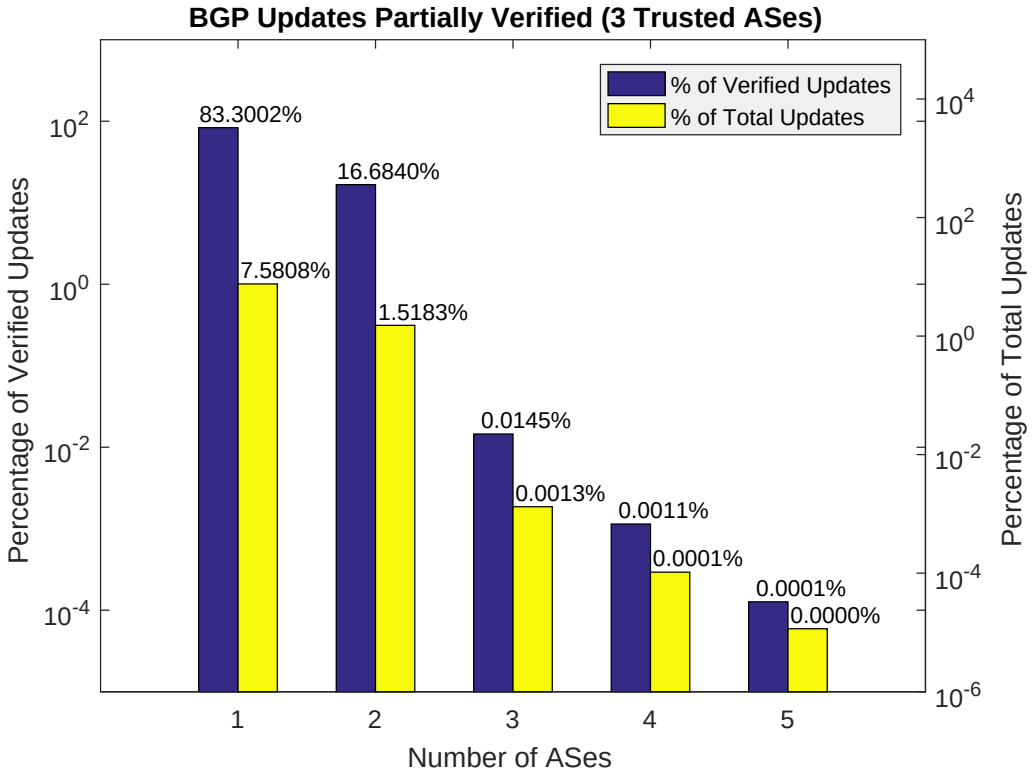


Figure 5.39: Partially Verified BGP Updates - Dataset 3 - 3 Trusted ASes

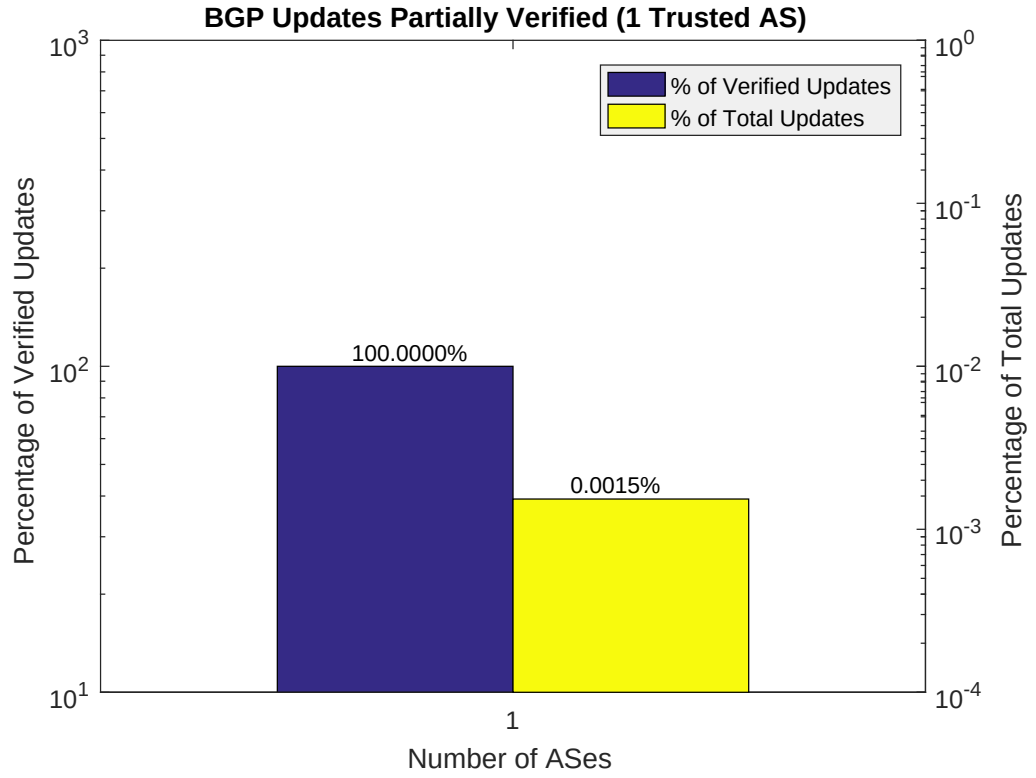


Figure 5.40: Partially Verified BGP Updates - Dataset 4 - 1 Trusted AS

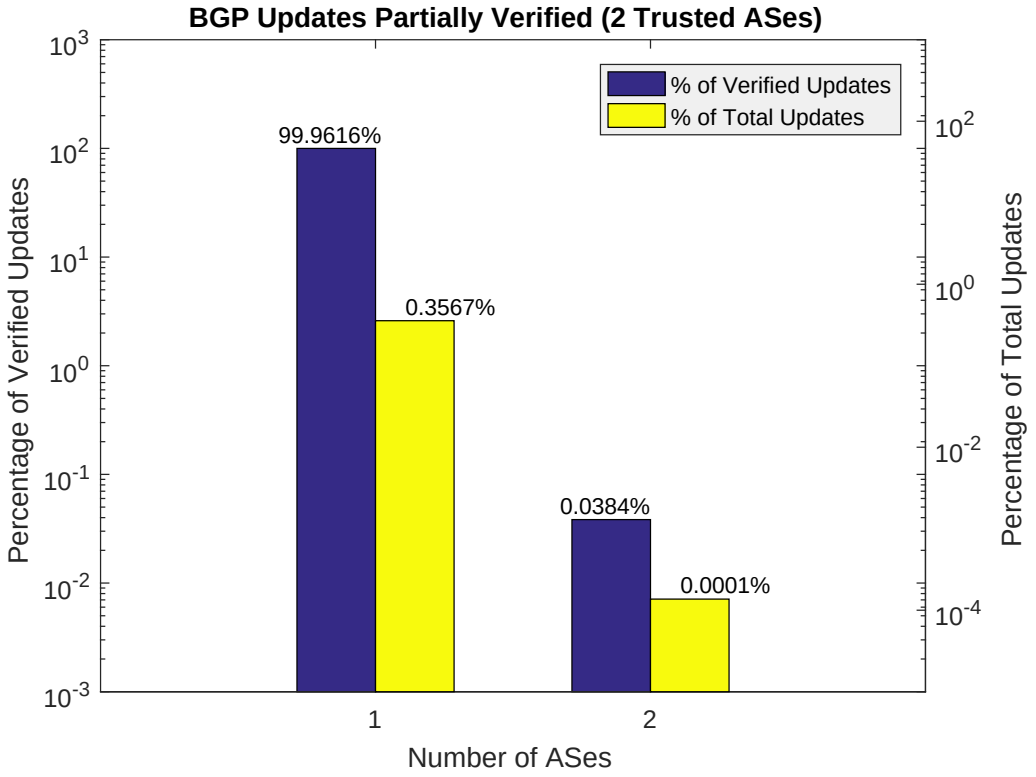


Figure 5.41: Partially Verified BGP Updates - Dataset 4 - 2 Trusted ASes

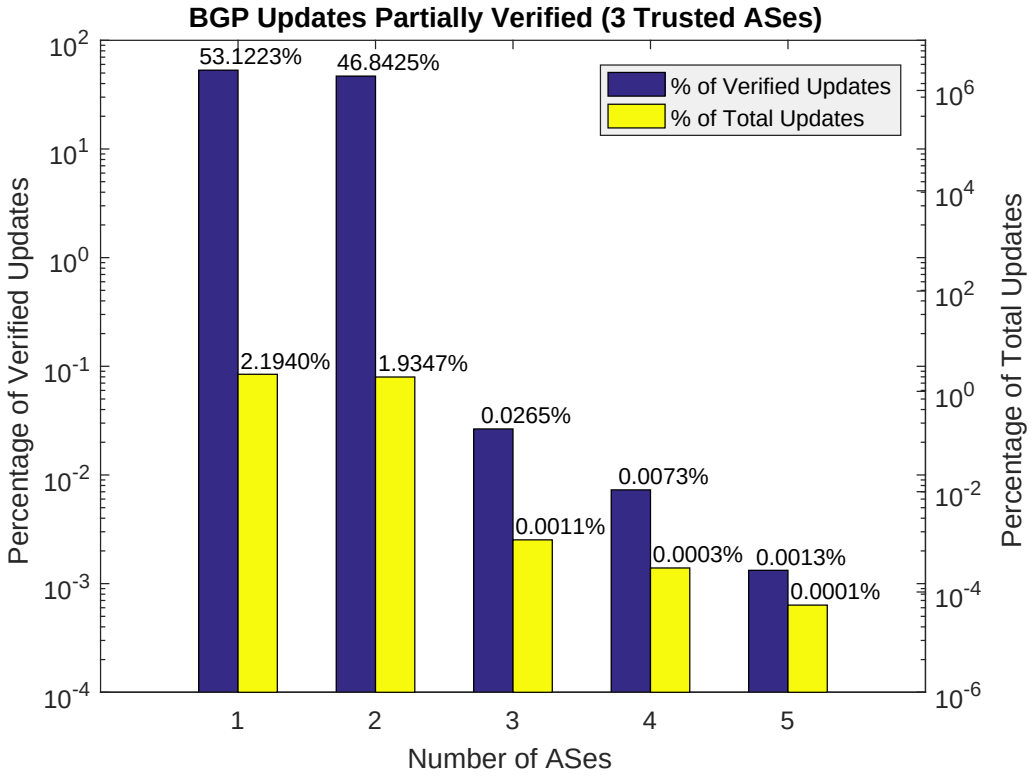


Figure 5.42: Partially Verified BGP Updates - Dataset 4 - 3 Trusted ASes

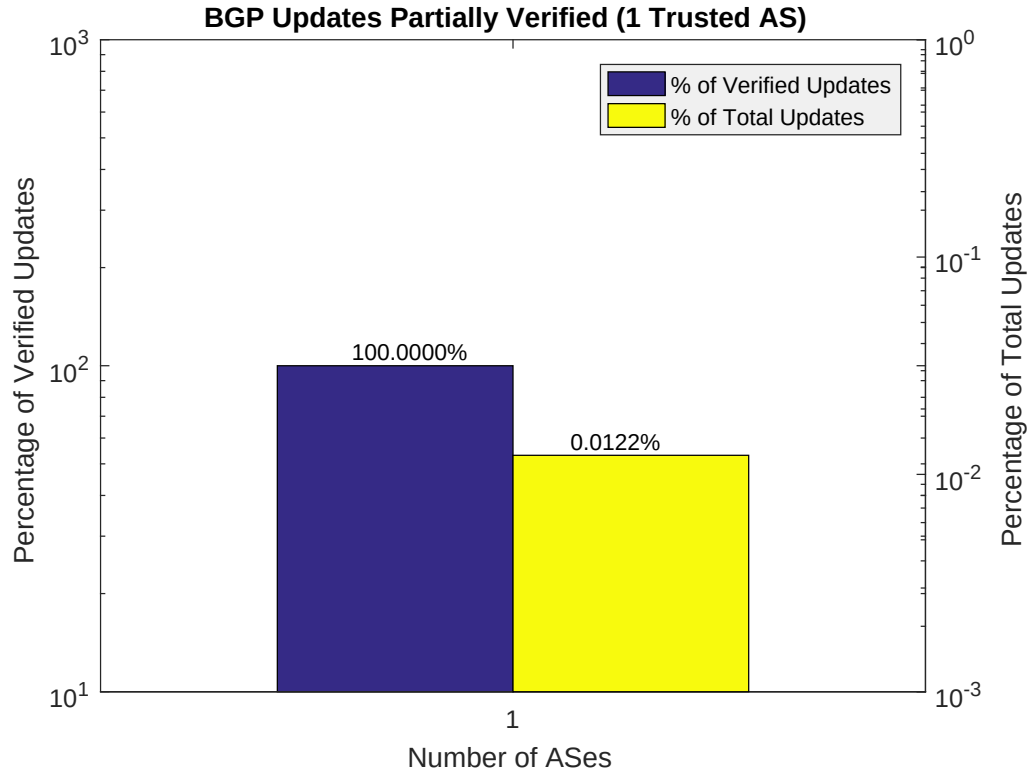


Figure 5.43: Partially Verified BGP Updates - Dataset 5 - 1 Trusted AS

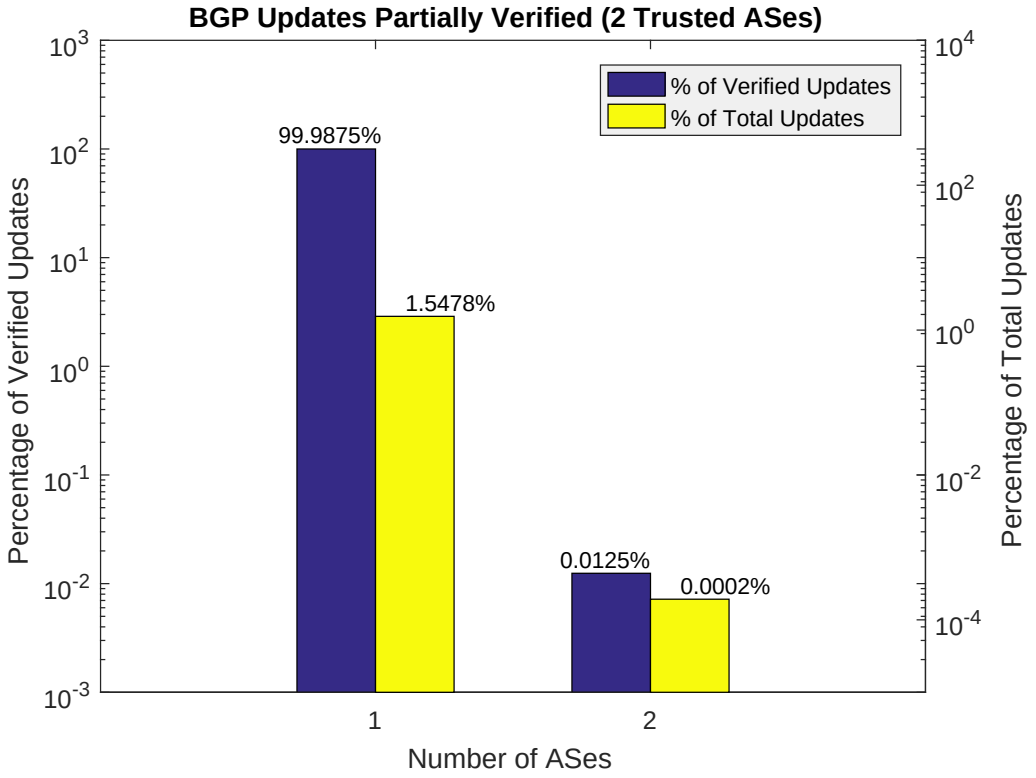


Figure 5.44: Partially Verified BGP Updates - Dataset 5 - 2 Trusted ASes

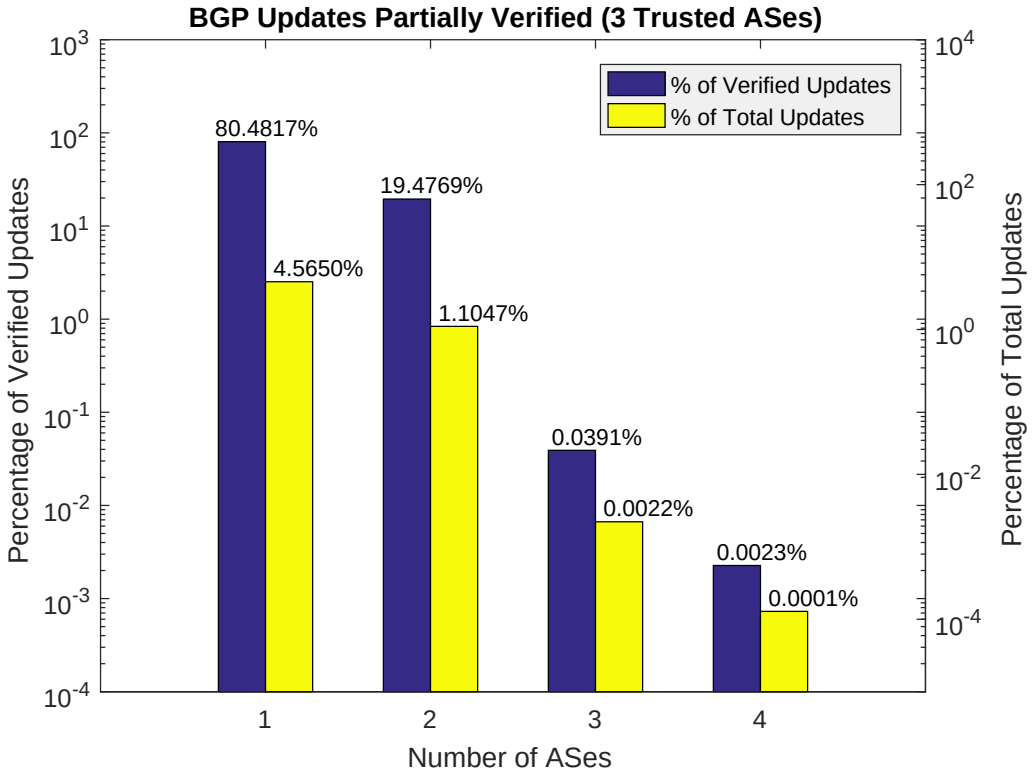


Figure 5.45: Partially Verified BGP Updates - Dataset 5 - 3 Trusted ASes

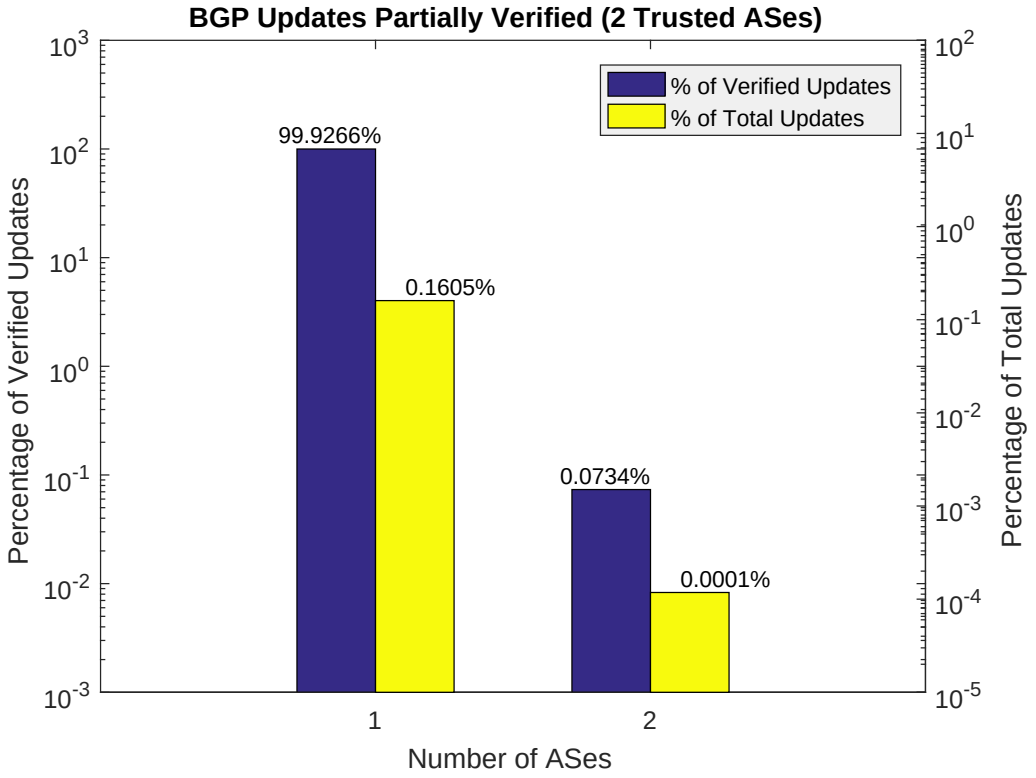


Figure 5.46: Partially Verified BGP Updates - Dataset 6 - 2 Trusted ASes

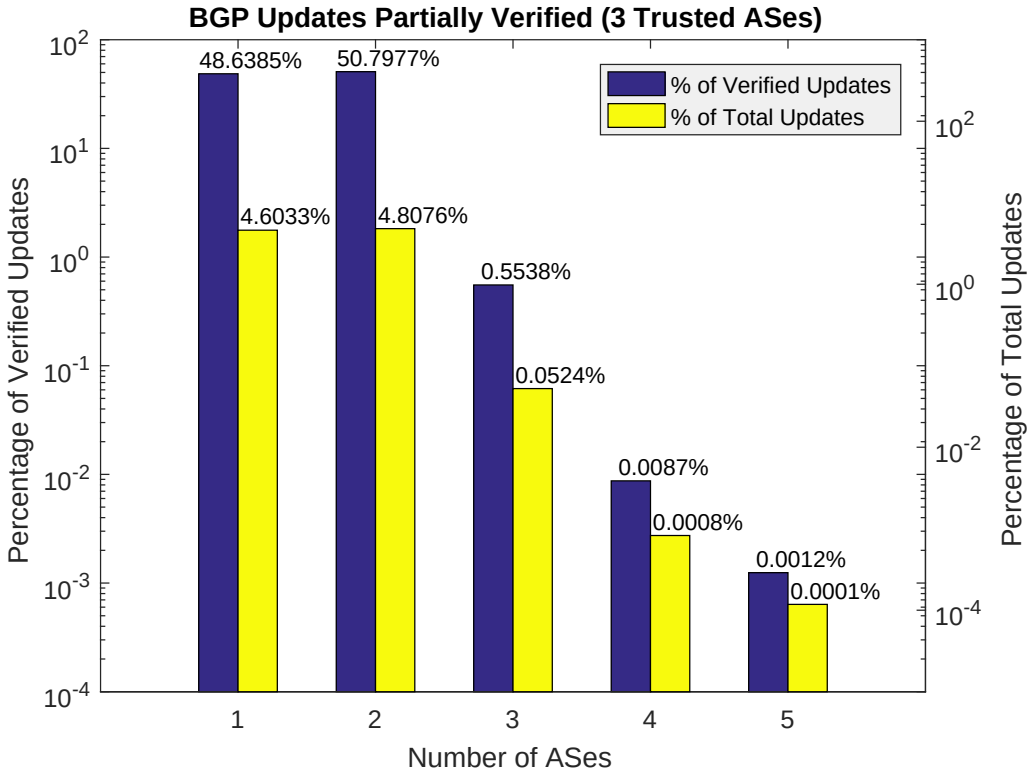


Figure 5.47: Partially Verified BGP Updates - Dataset 6 - 3 Trusted ASes

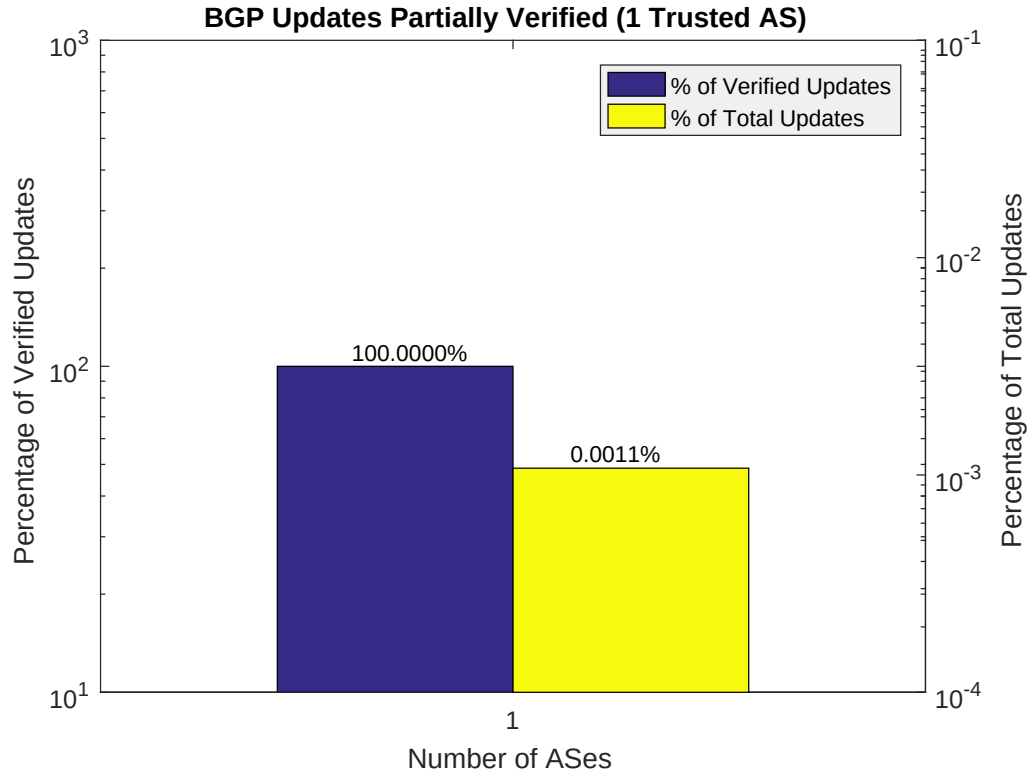


Figure 5.48: Partially Verified BGP Updates - Dataset 7 - 1 Trusted AS

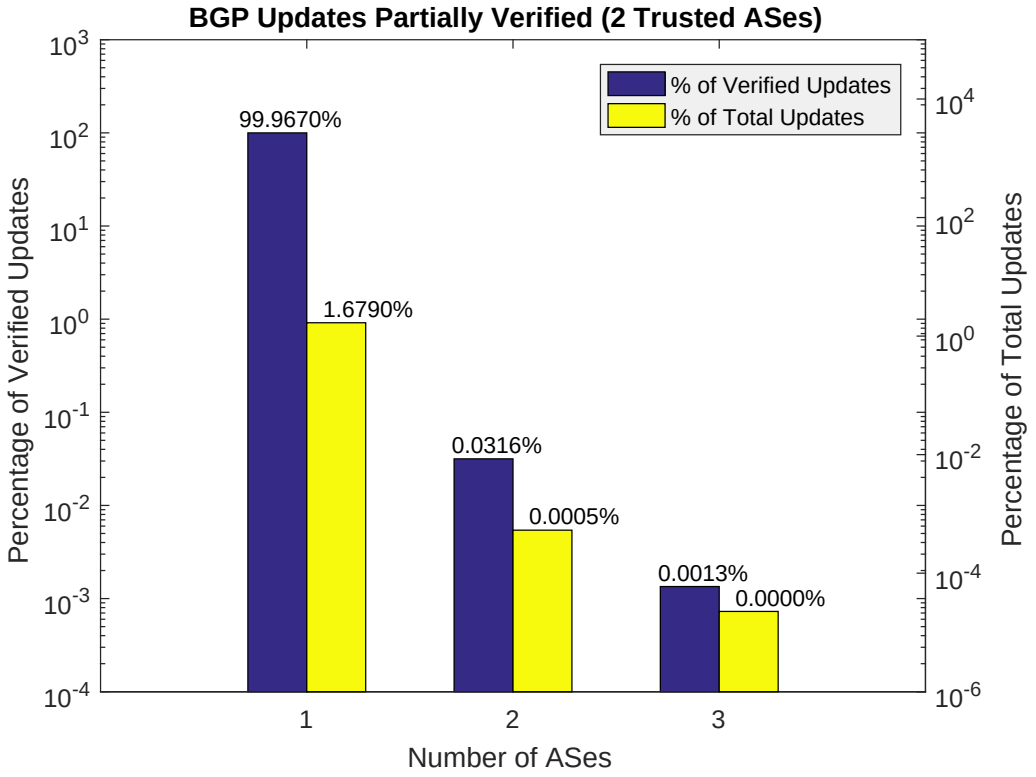


Figure 5.49: Partially Verified BGP Updates - Dataset 7 - 2 Trusted ASes

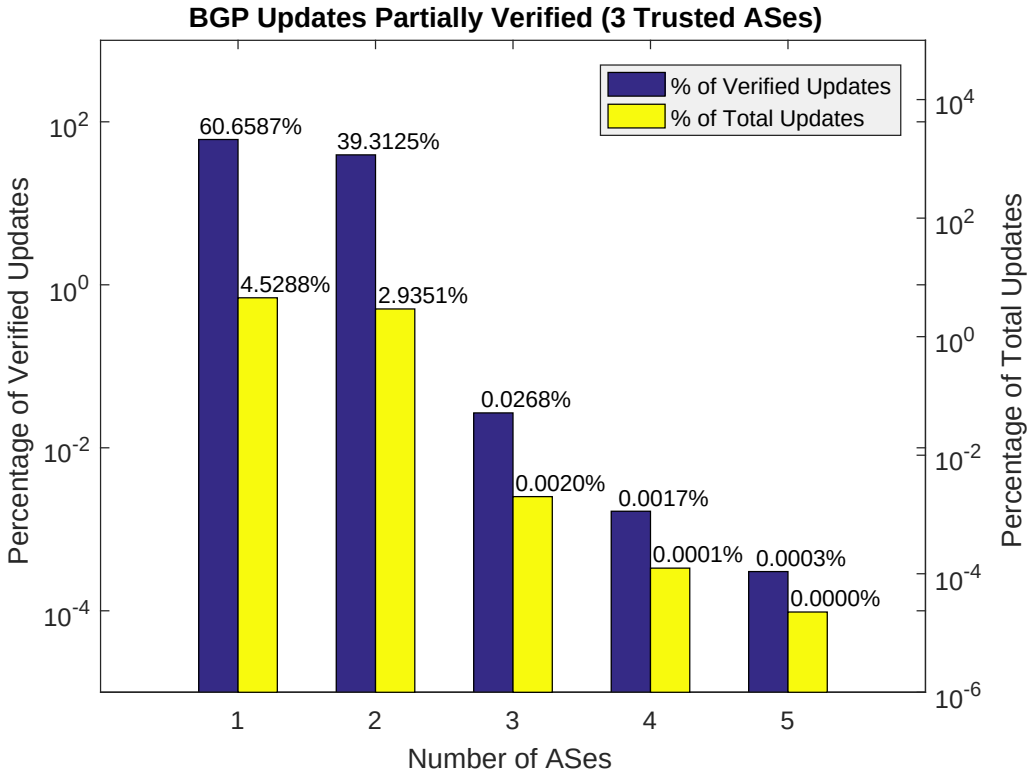


Figure 5.50: Partially Verified BGP Updates - Dataset 7 - 3 Trusted ASes

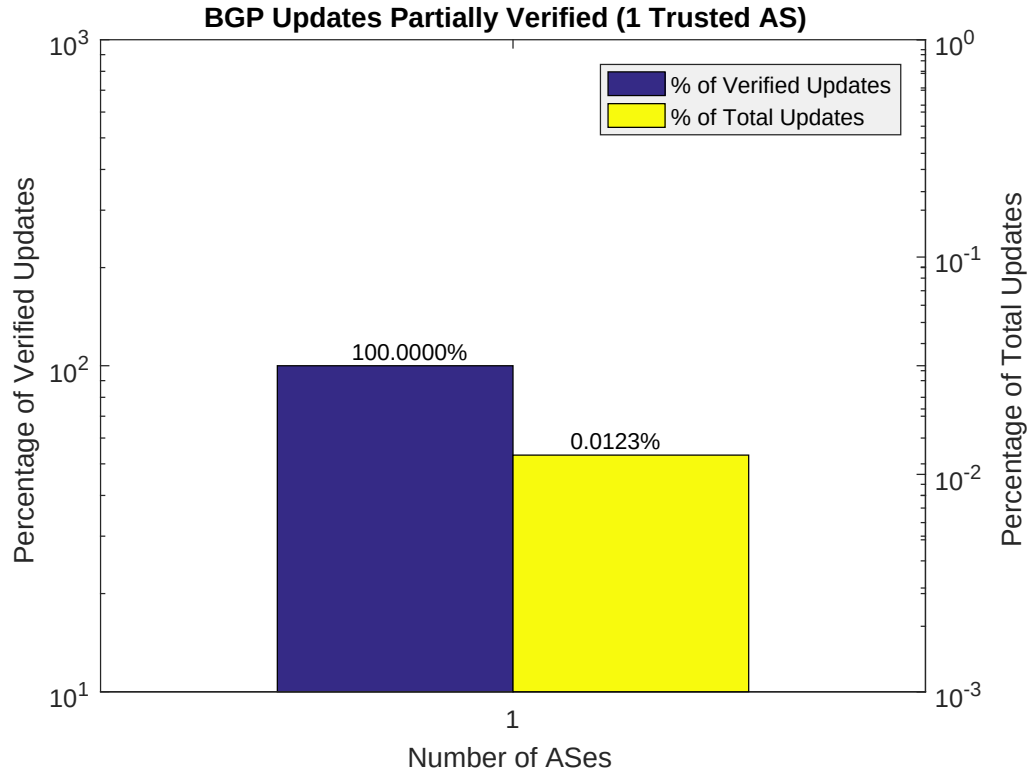


Figure 5.51: Partially Verified BGP Updates - Dataset 8 - 1 Trusted AS

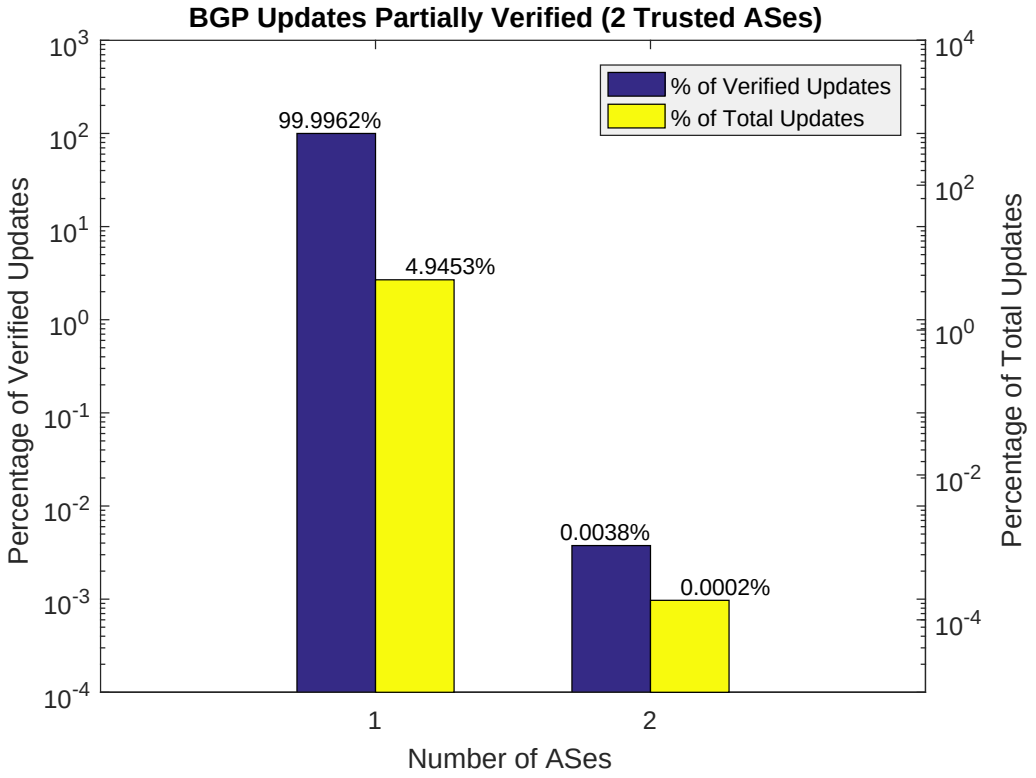


Figure 5.52: Partially Verified BGP Updates - Dataset 8 - 2 Trusted ASes

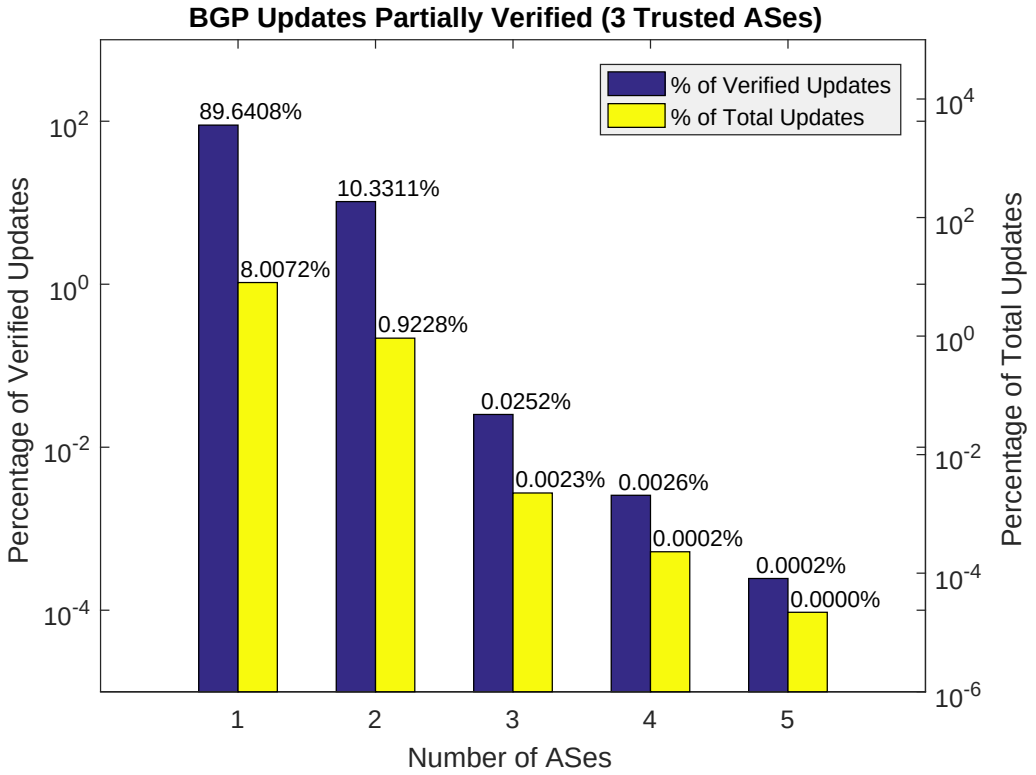


Figure 5.53: Partially Verified BGP Updates - Dataset 8 - 3 Trusted ASes

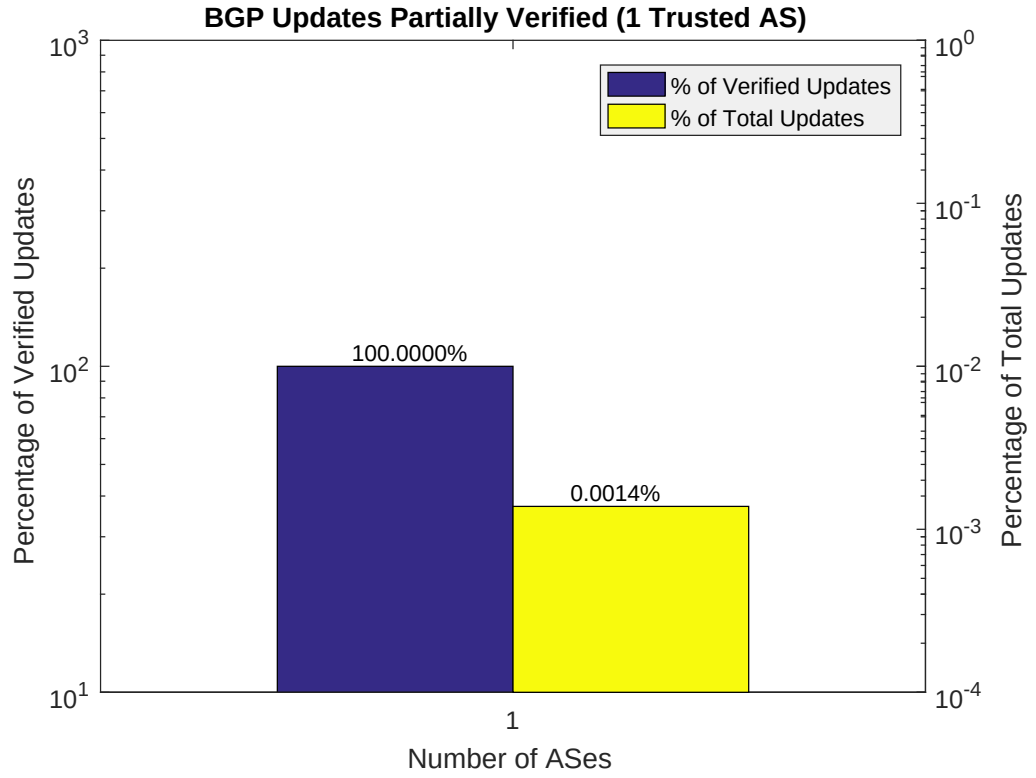


Figure 5.54: Partially Verified BGP Updates - Dataset 9 - 1 Trusted AS

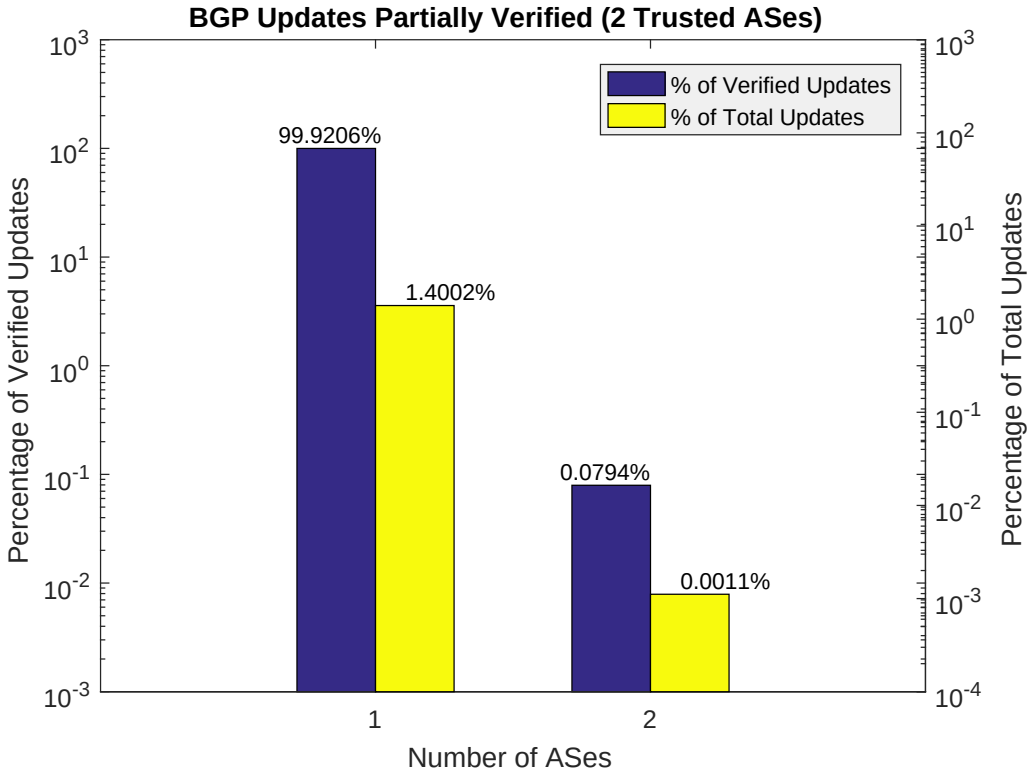


Figure 5.55: Partially Verified BGP Updates - Dataset 9 - 2 Trusted ASes

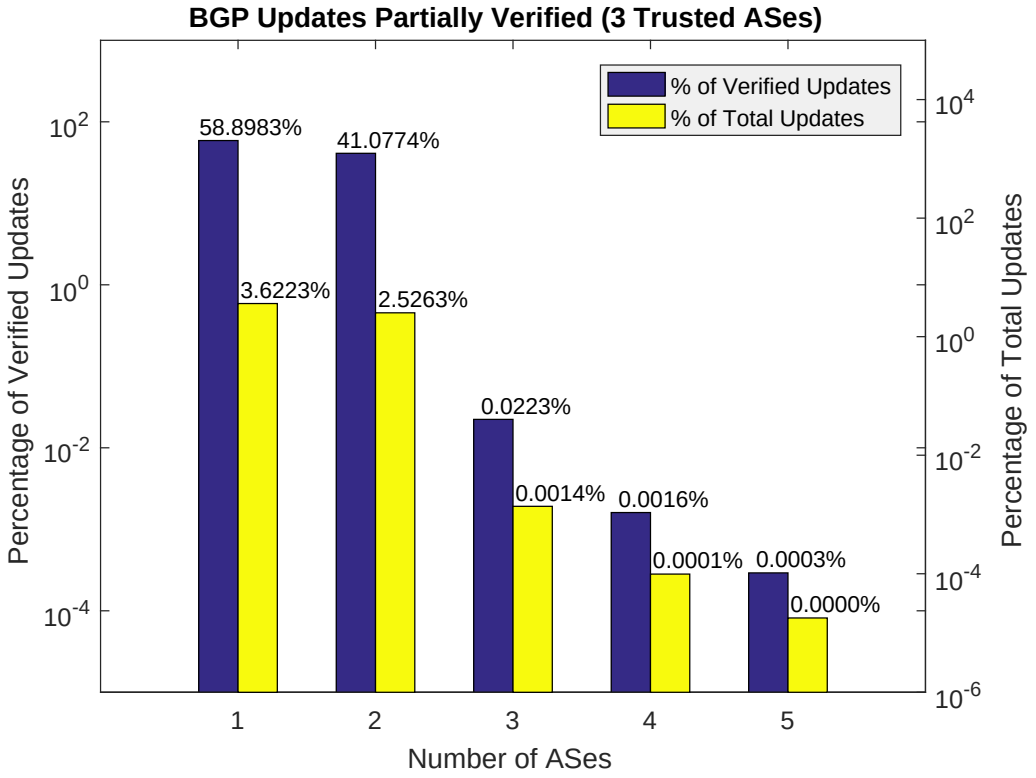


Figure 5.56: Partially Verified BGP Updates - Dataset 9 - 3 Trusted ASes

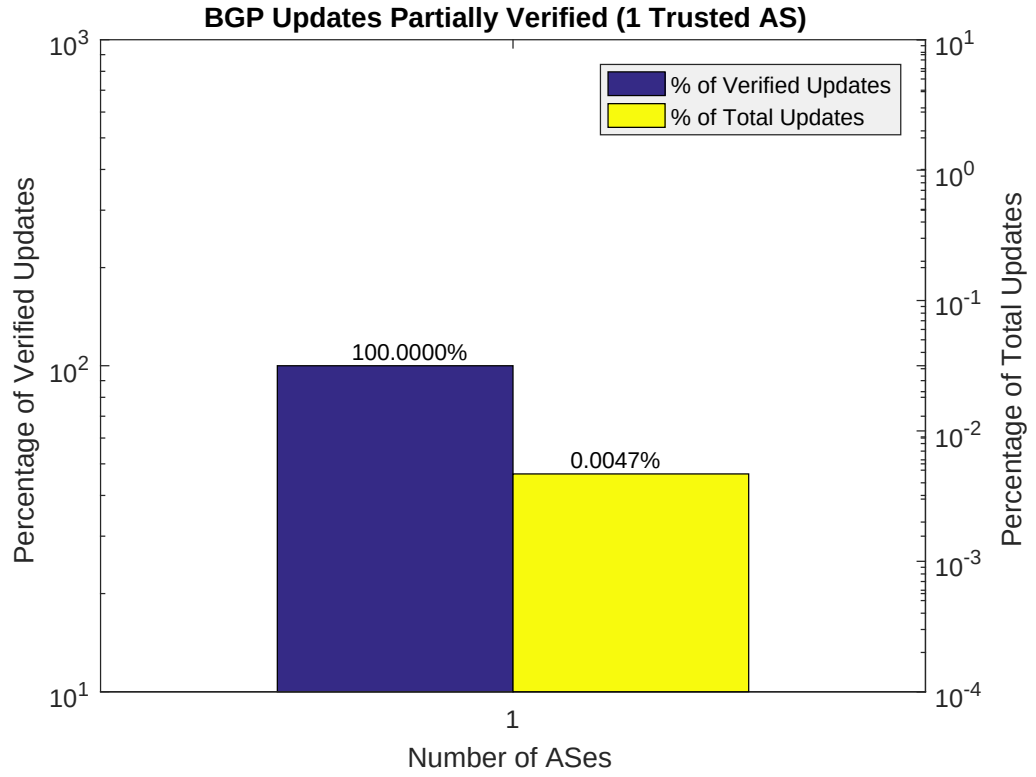


Figure 5.57: Partially Verified BGP Updates - Dataset 10 - 1 Trusted AS

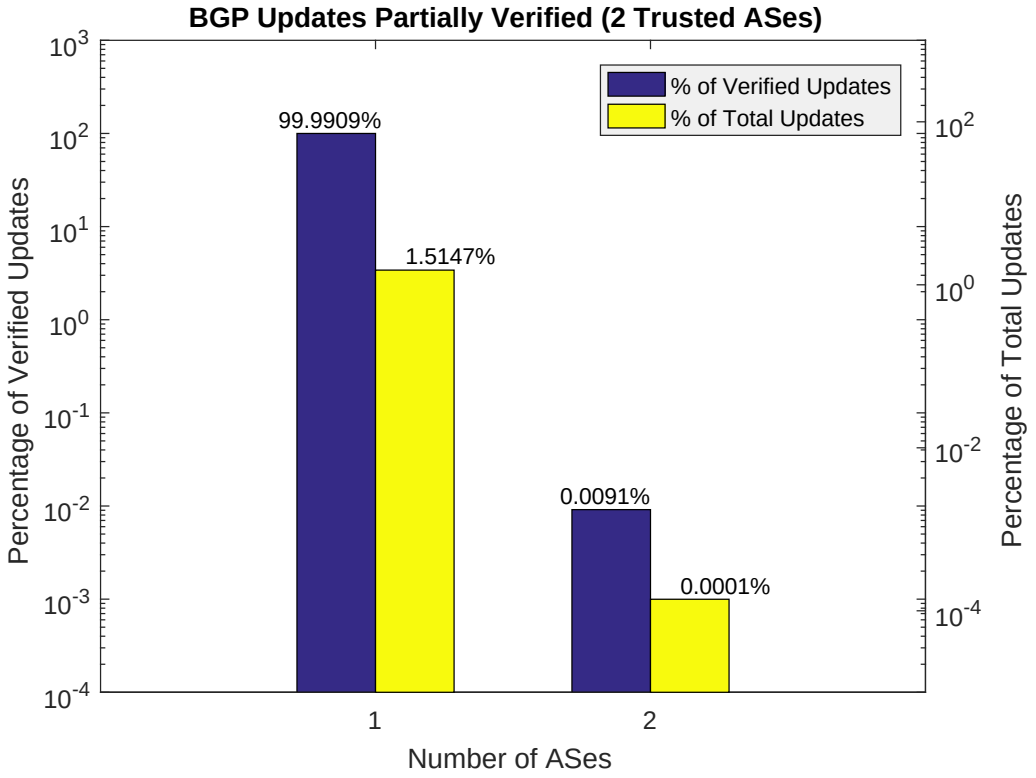


Figure 5.58: Partially Verified BGP Updates - Dataset 10 - 2 Trusted ASes

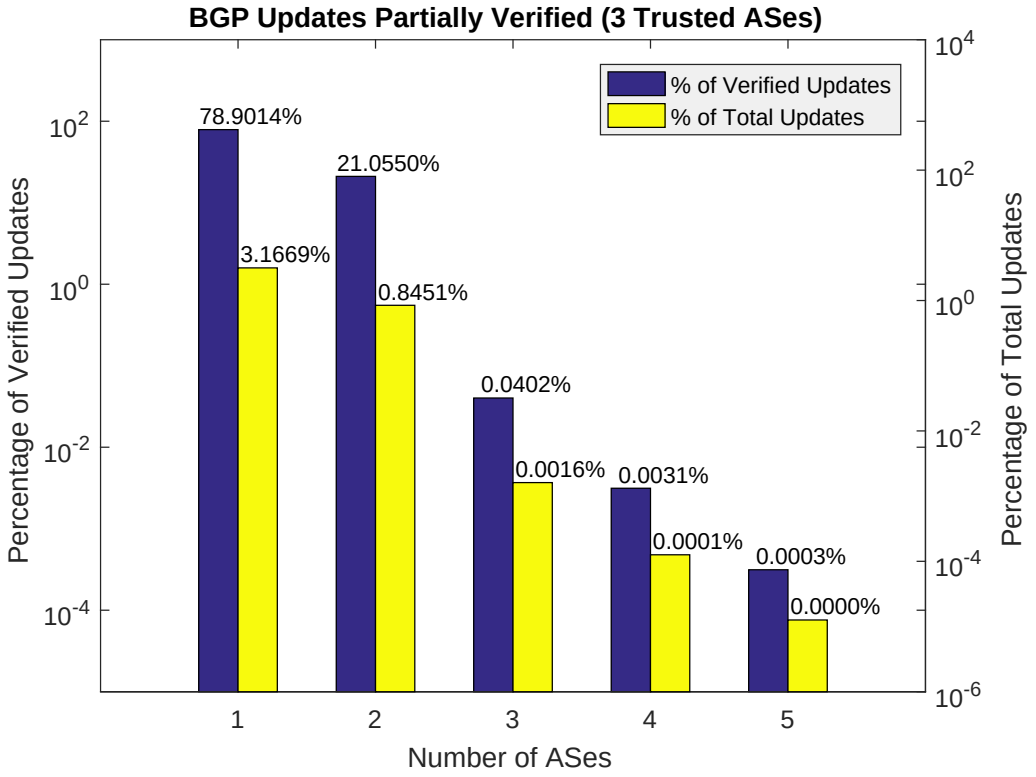


Figure 5.59: Partially Verified BGP Updates - Dataset 10 - 3 Trusted ASes

Data on Totally Verified BGP Updates

The subsequent graphs present the raw data regarding the BGP Updates that are totally verified by others already stored on the Blockchain. This data is in percentages. The blue bars represent the percentages in relation to the amount of updates that were verified. The yellow bars represent the percentages in relation to the total updates in the dataset utilized.

Each graph corresponds to a different dataset, which was executed three times, when considering 1, 2, or 3 ASes as initially trusted by the algorithm.

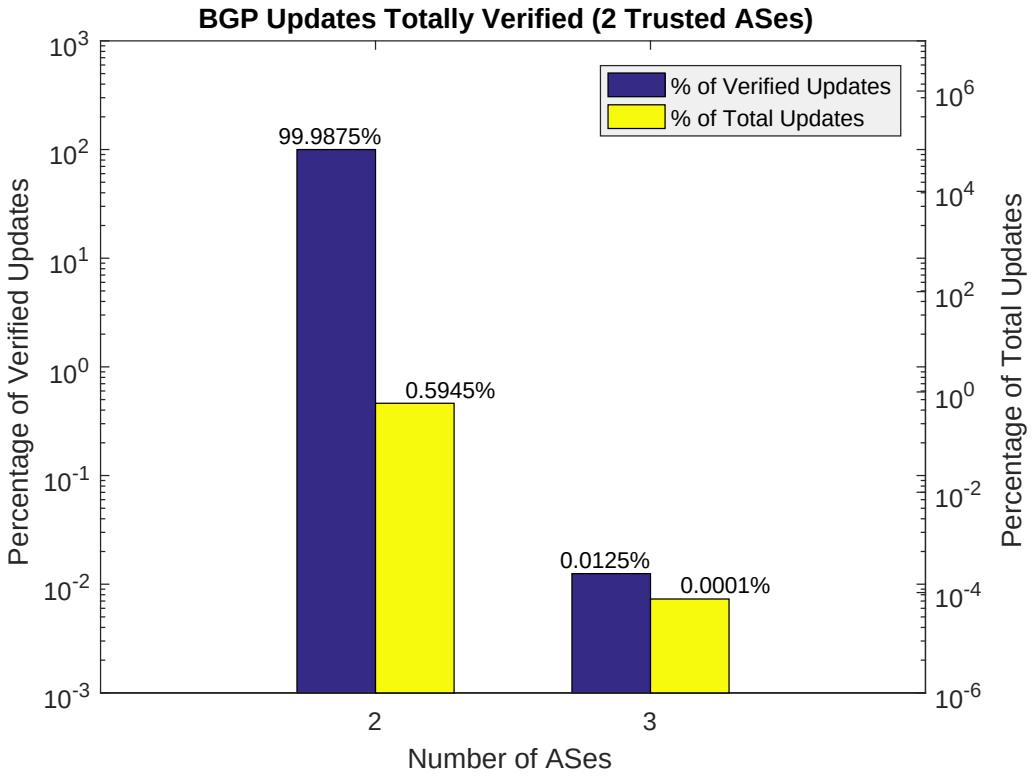


Figure 5.60: Totally Verified BGP Updates - Dataset 1 - 2 Trusted ASes

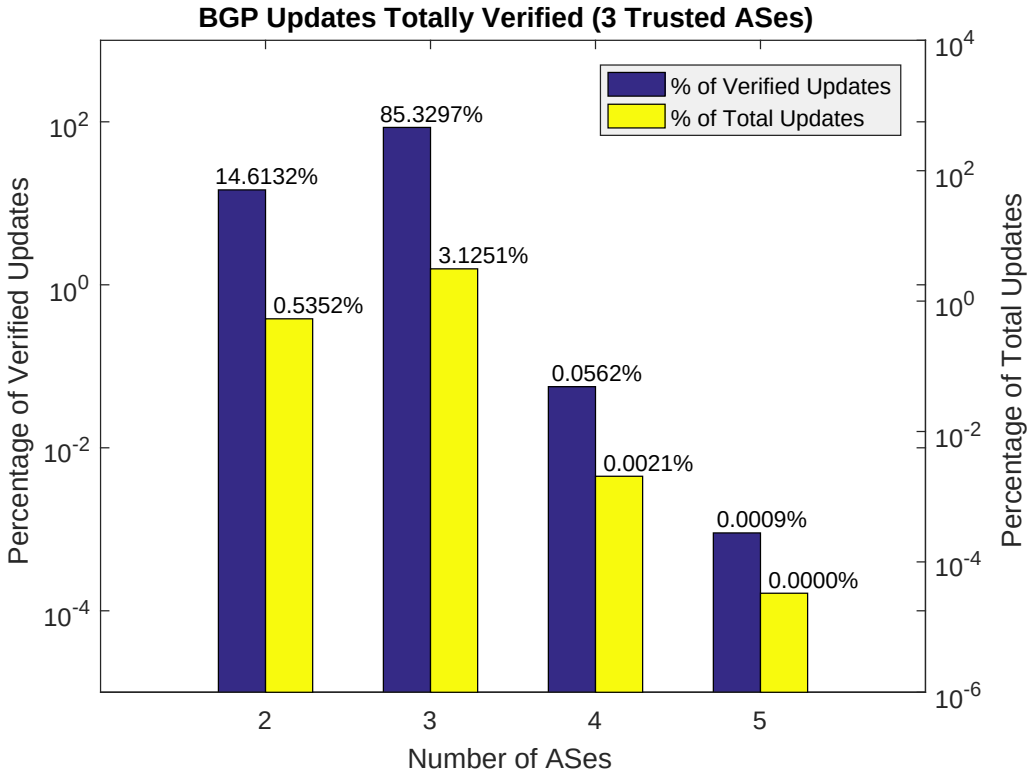


Figure 5.61: Totally Verified BGP Updates - Dataset 1 - 3 Trusted ASes

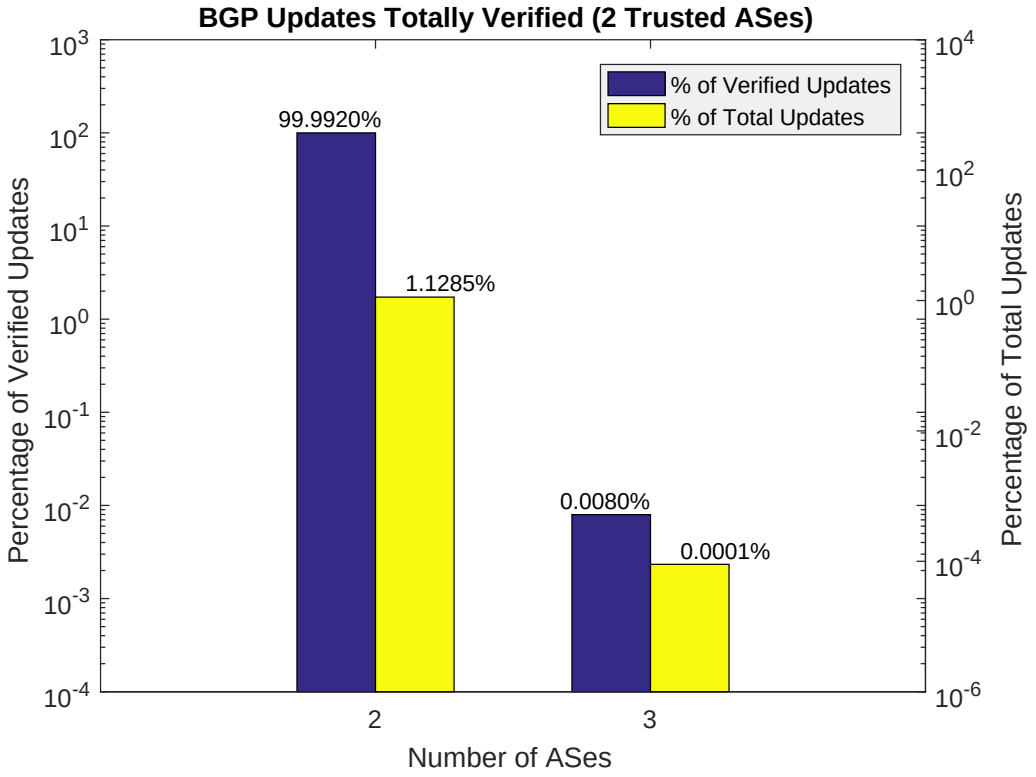


Figure 5.62: Totally Verified BGP Updates - Dataset 2 - 2 Trusted ASes

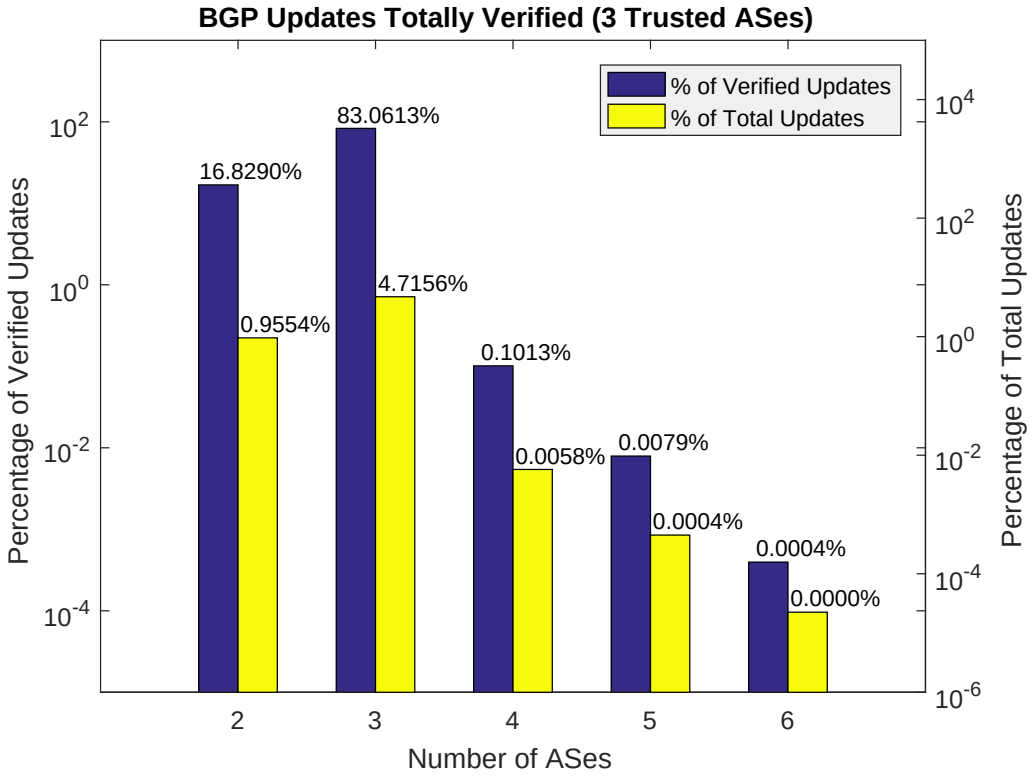


Figure 5.63: Totally Verified BGP Updates - Dataset 2 - 3 Trusted ASes

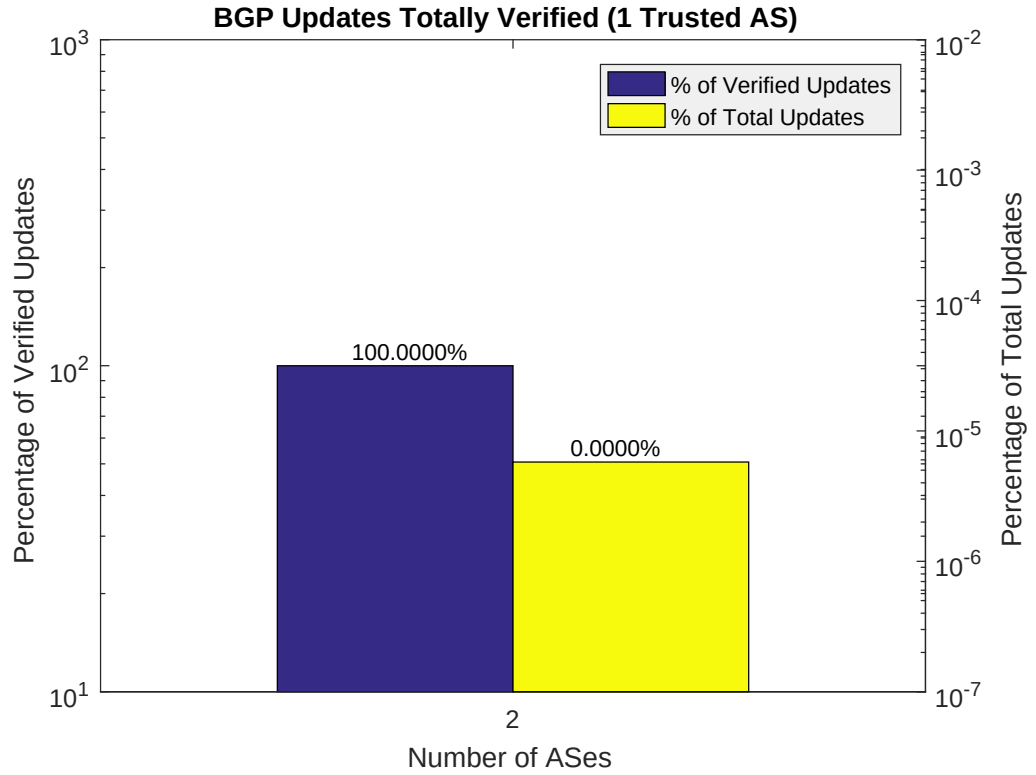


Figure 5.64: Totally Verified BGP Updates - Dataset 3 - 1 Trusted AS

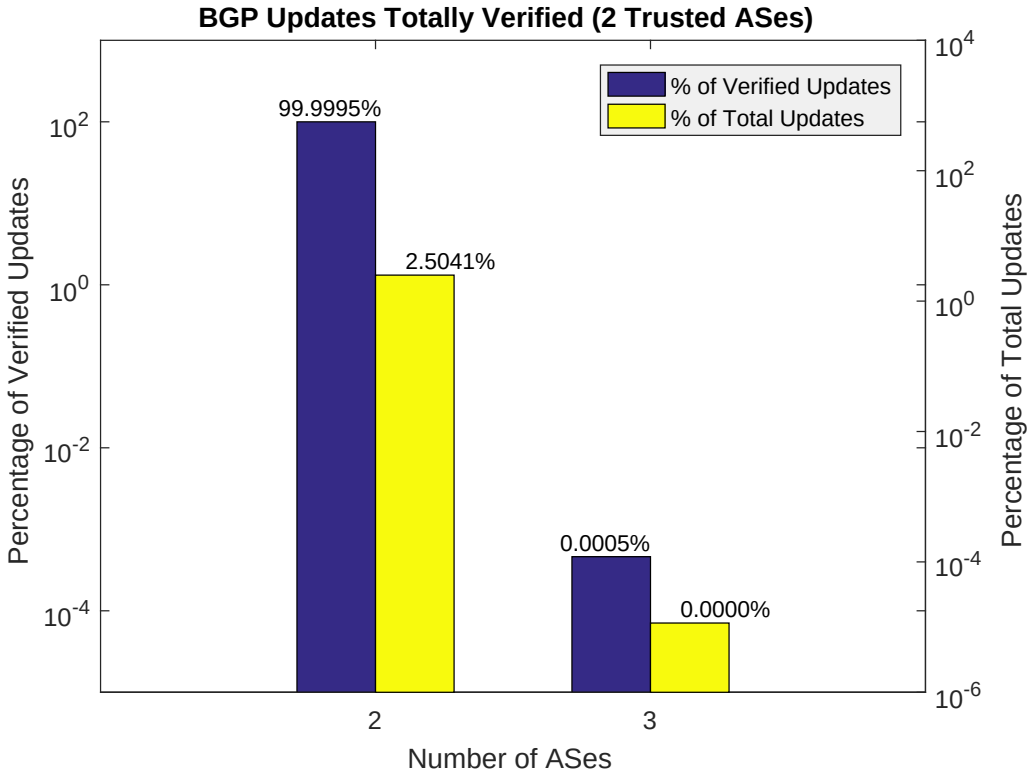


Figure 5.65: Totally Verified BGP Updates - Dataset 3 - 2 Trusted ASes

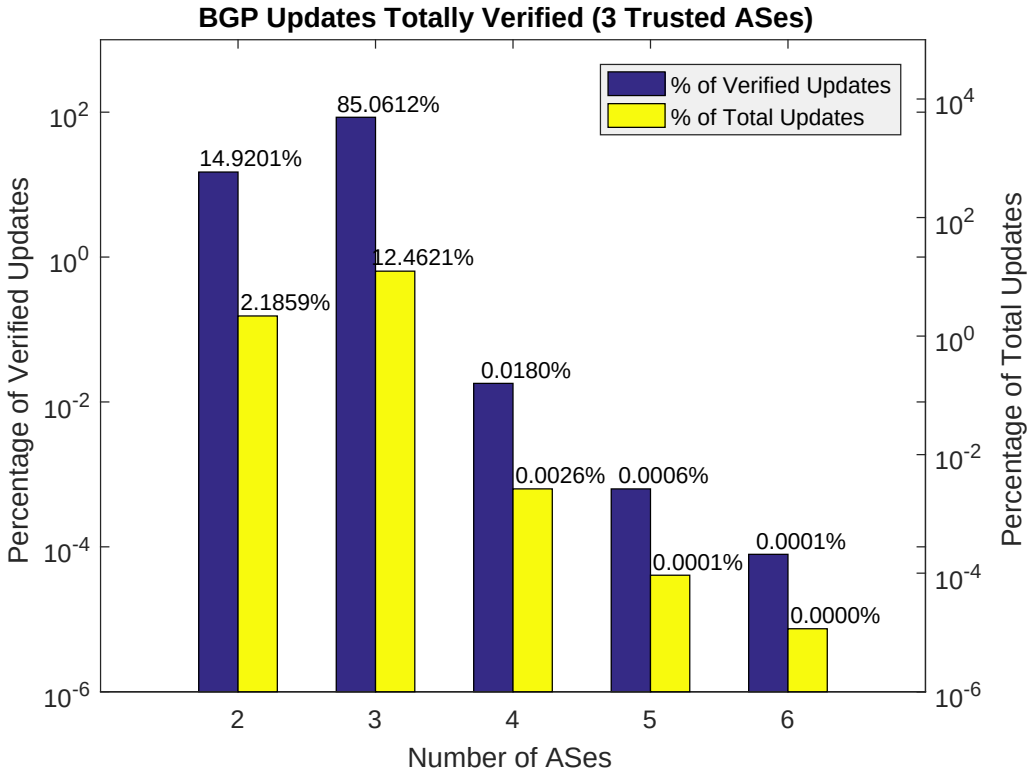


Figure 5.66: Totally Verified BGP Updates - Dataset 3 - 3 Trusted ASes

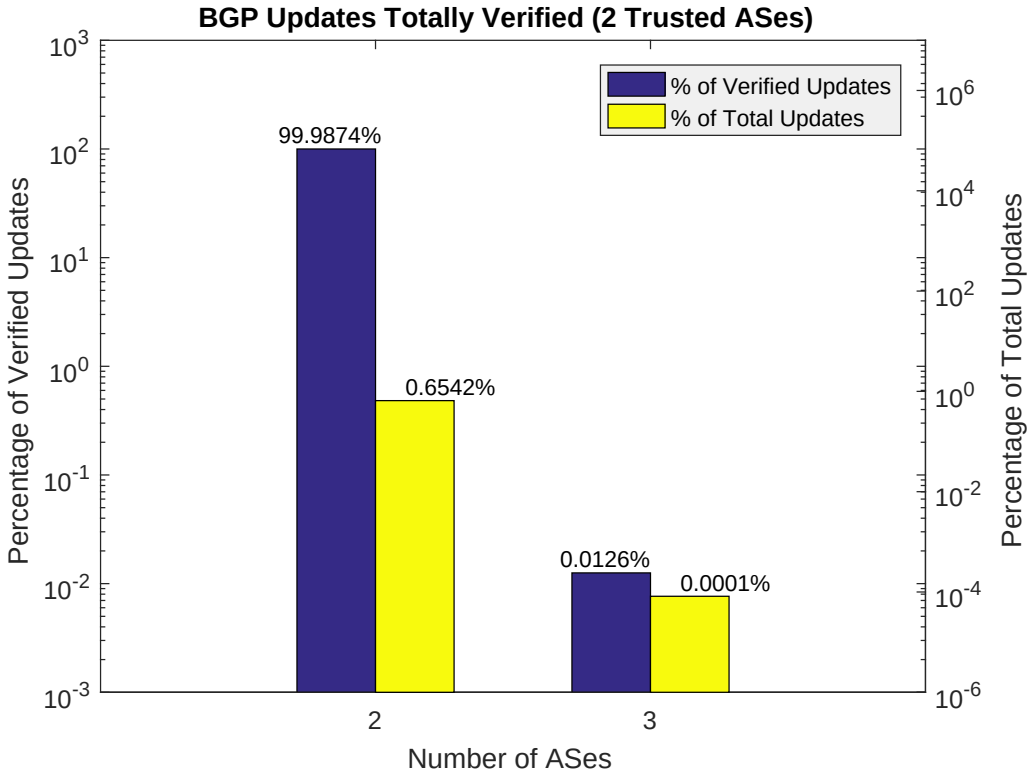


Figure 5.67: Totally Verified BGP Updates - Dataset 4 - 2 Trusted ASes

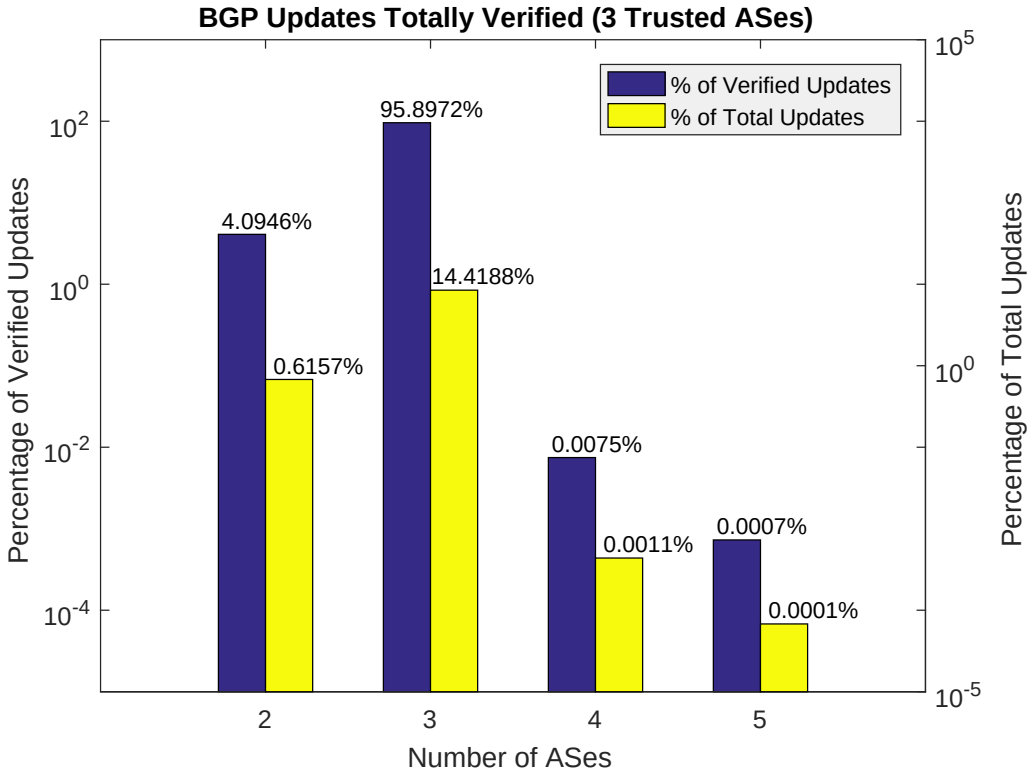


Figure 5.68: Totally Verified BGP Updates - Dataset 4 - 3 Trusted ASes

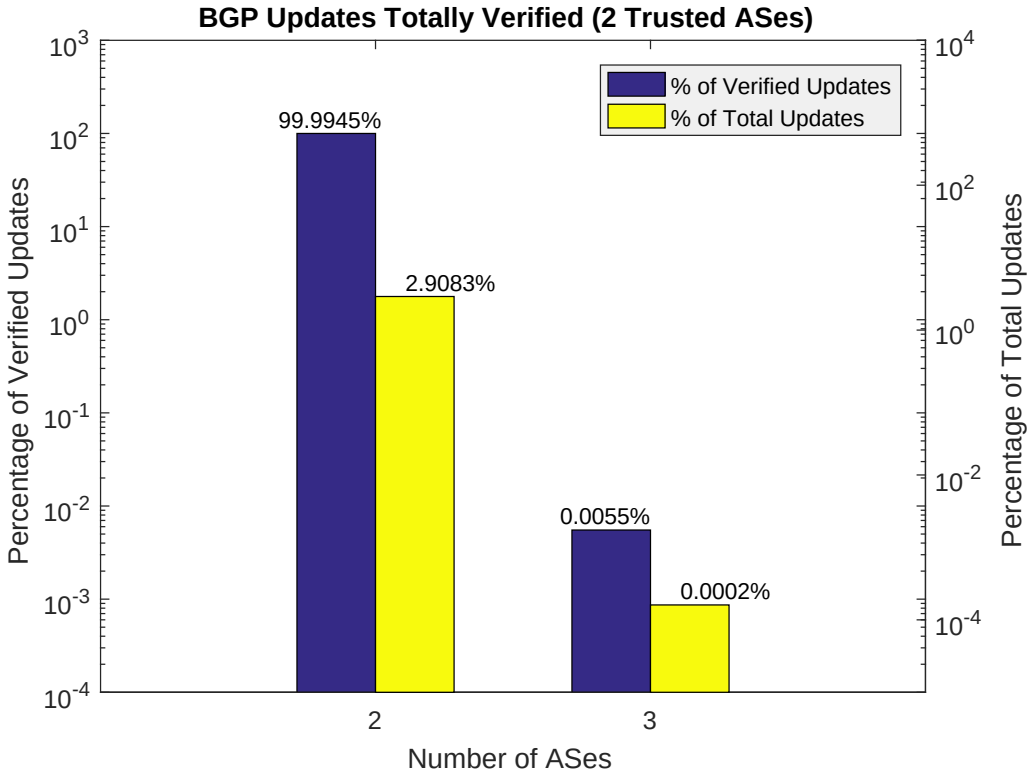


Figure 5.69: Totally Verified BGP Updates - Dataset 5 - 2 Trusted ASes

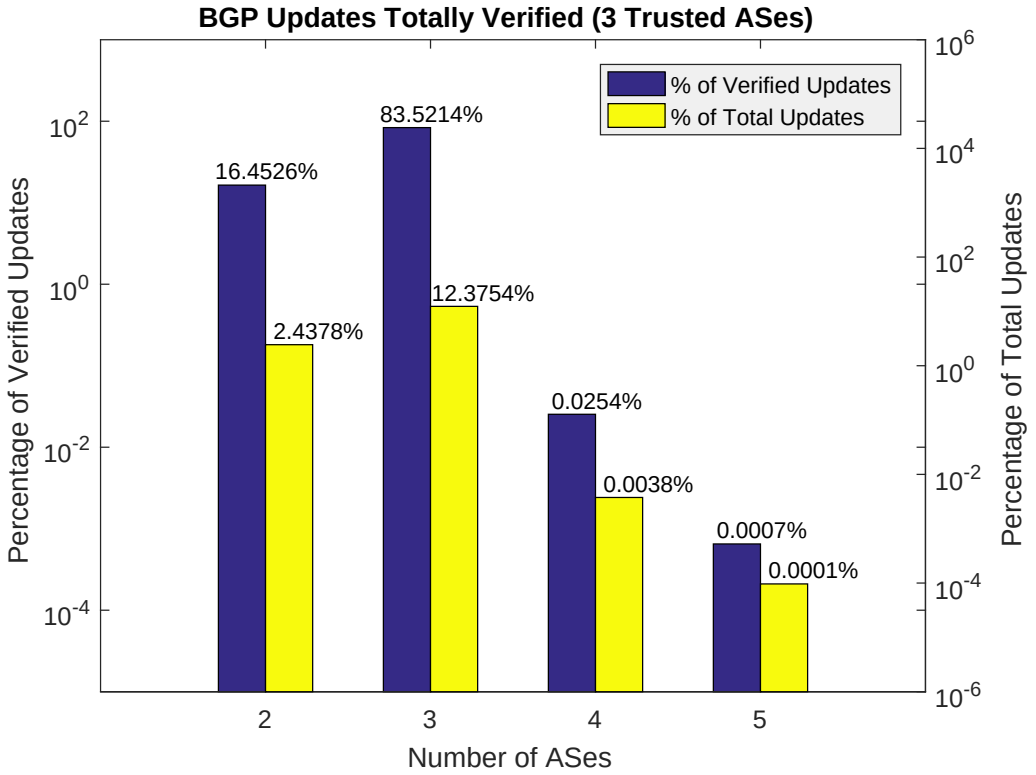


Figure 5.70: Totally Verified BGP Updates - Dataset 5 - 3 Trusted ASes

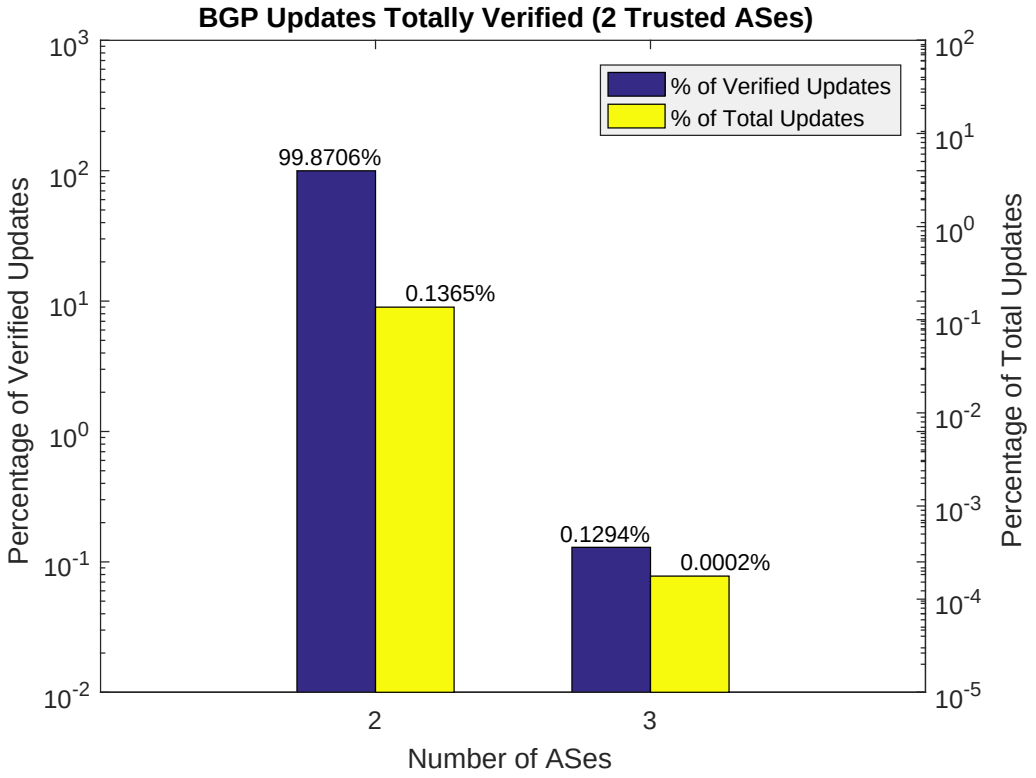


Figure 5.71: Totally Verified BGP Updates - Dataset 6 - 2 Trusted ASes

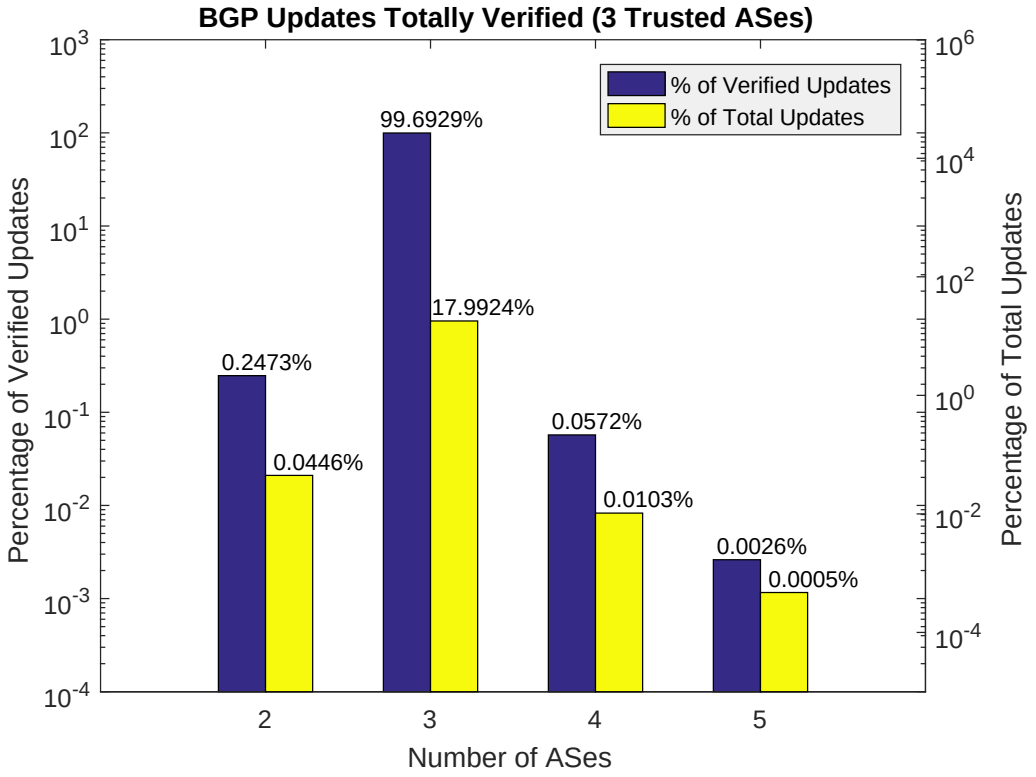


Figure 5.72: Totally Verified BGP Updates - Dataset 6 - 3 Trusted ASes

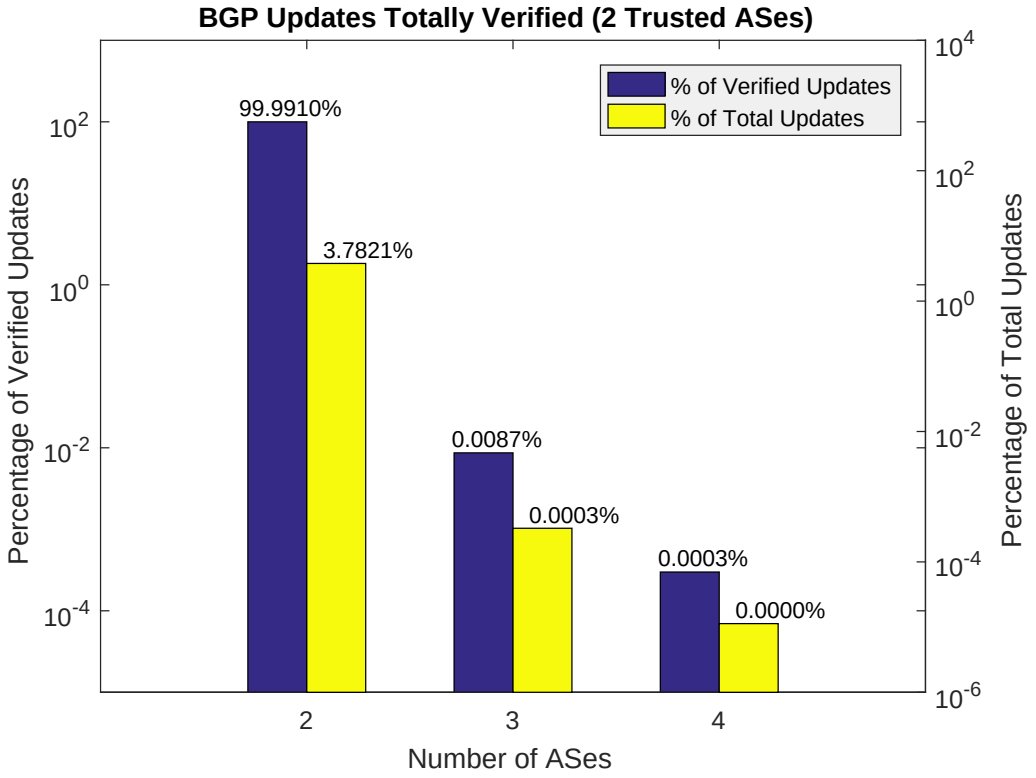


Figure 5.73: Totally Verified BGP Updates - Dataset 7 - 2 Trusted ASes

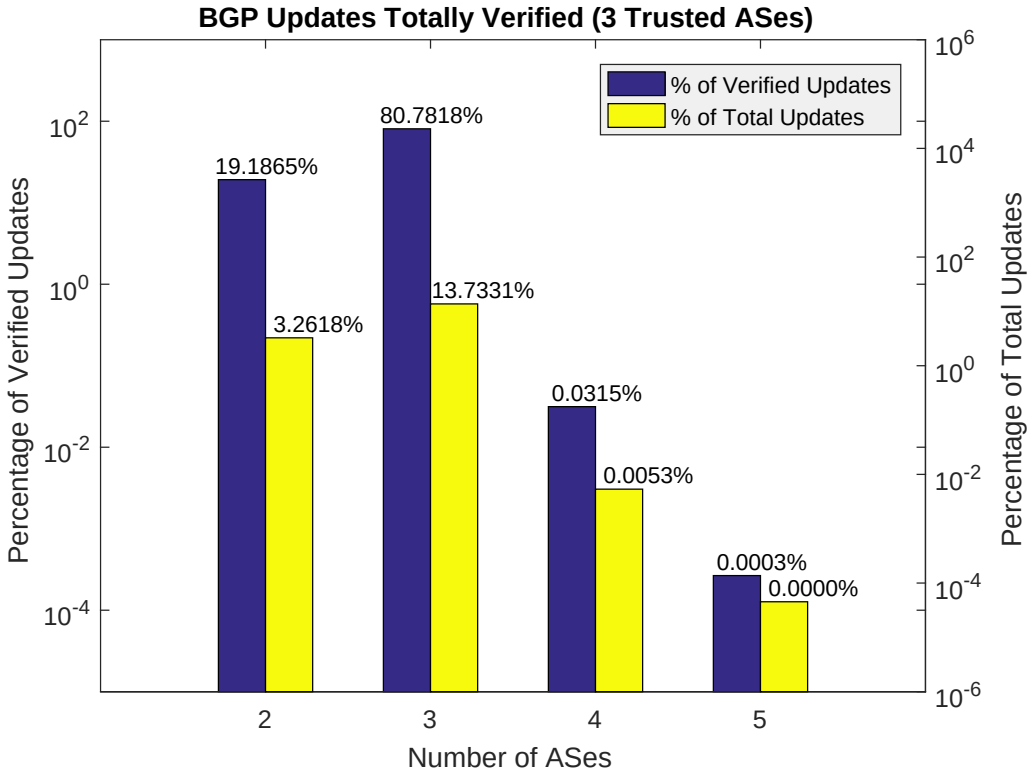


Figure 5.74: Totally Verified BGP Updates - Dataset 7 - 3 Trusted ASes

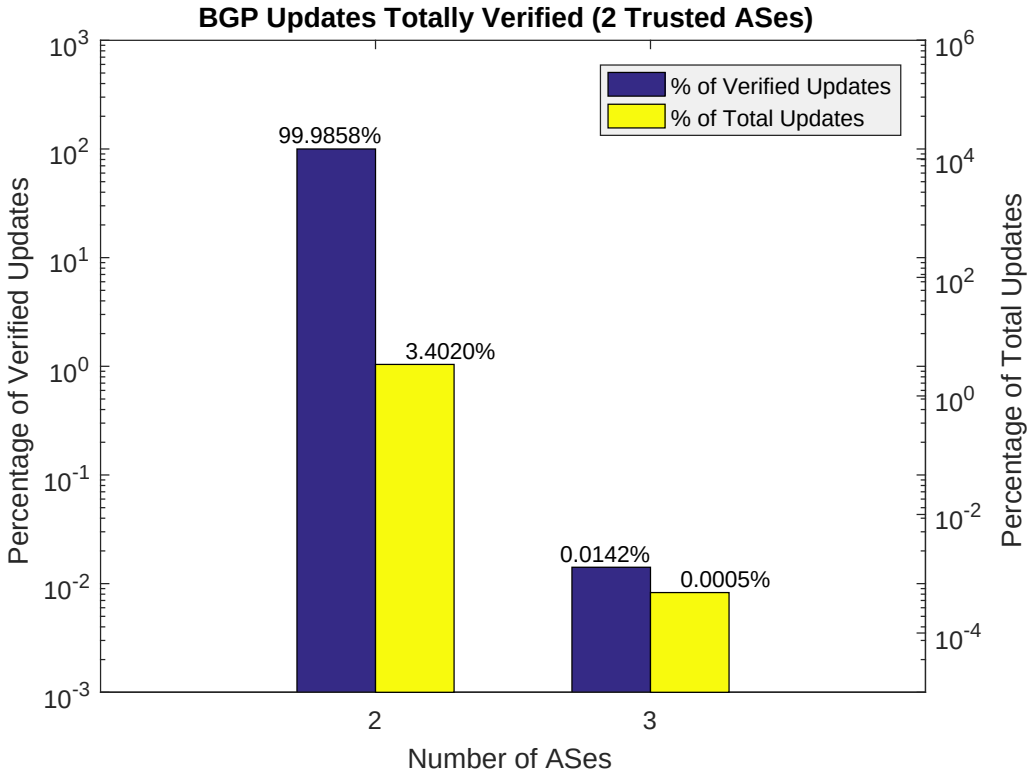


Figure 5.75: Totally Verified BGP Updates - Dataset 8 - 2 Trusted ASes

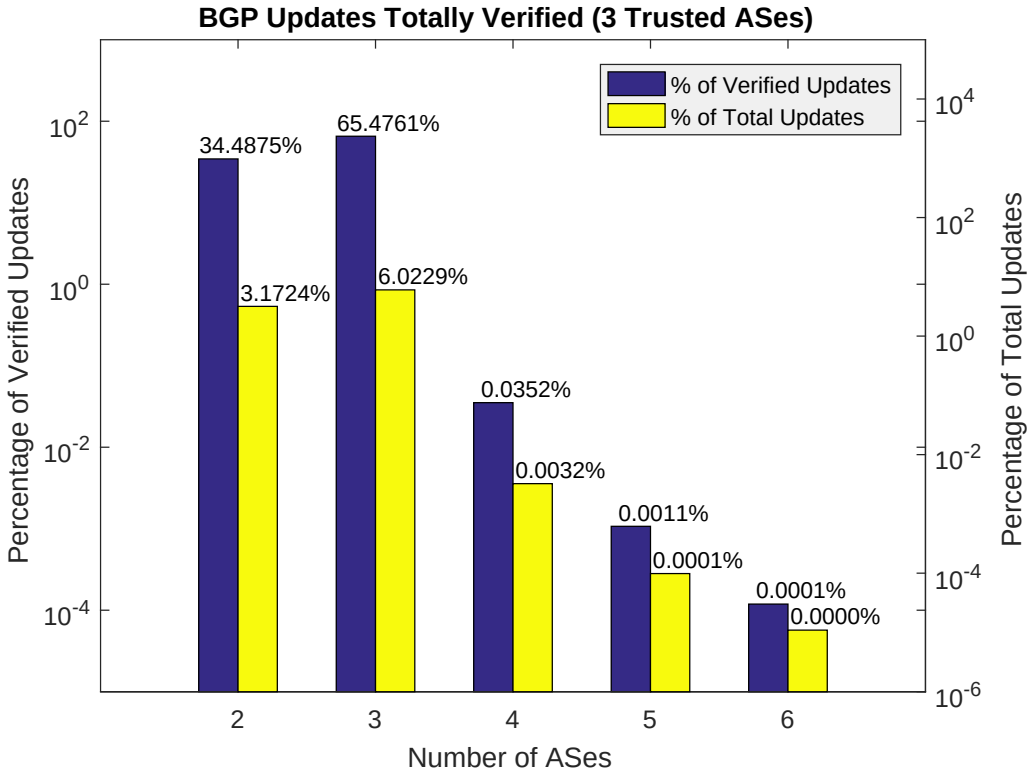


Figure 5.76: Totally Verified BGP Updates - Dataset 8 - 3 Trusted ASes

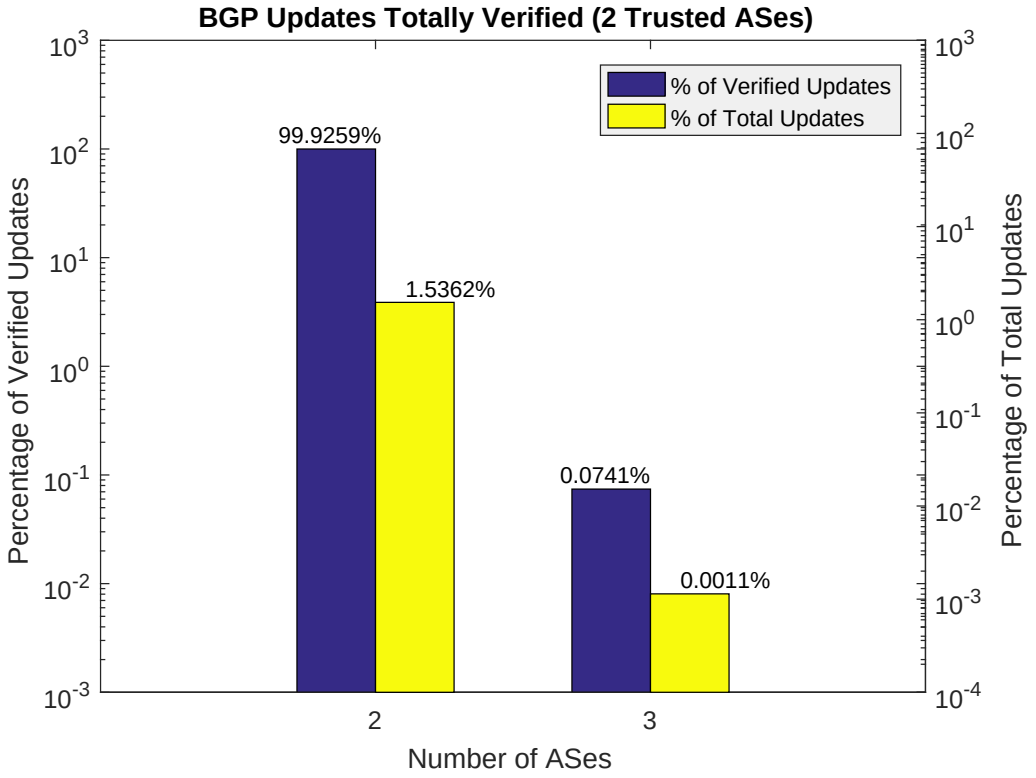


Figure 5.77: Totally Verified BGP Updates - Dataset 9 - 2 Trusted ASes

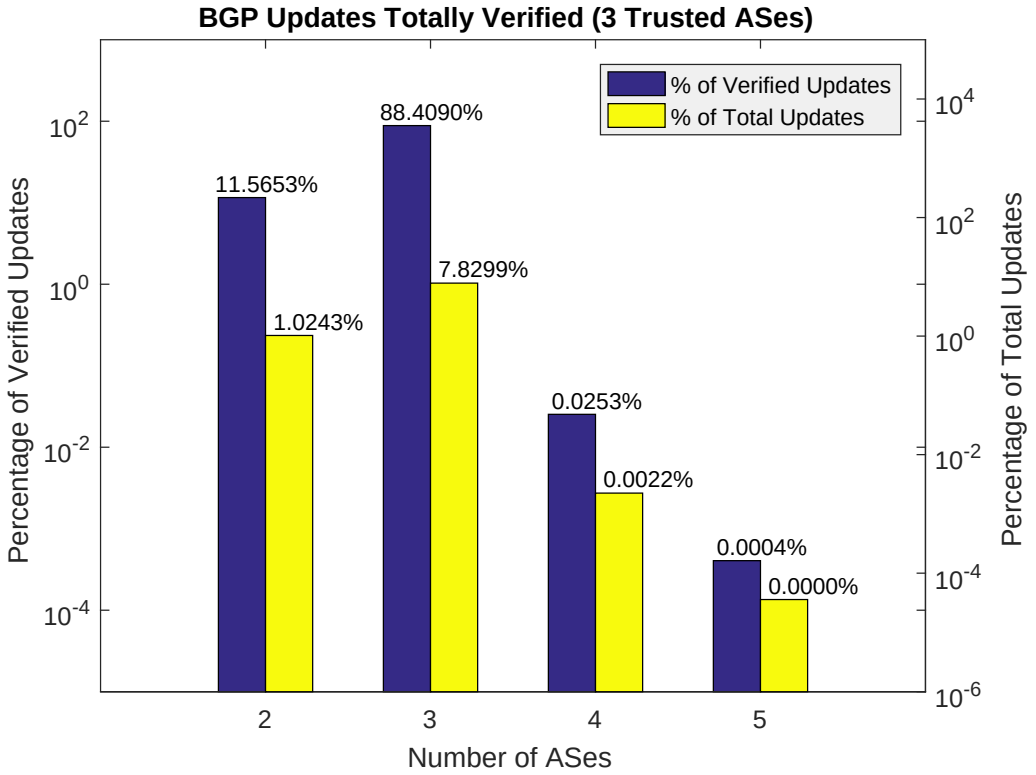


Figure 5.78: Totally Verified BGP Updates - Dataset 9 - 3 Trusted ASes

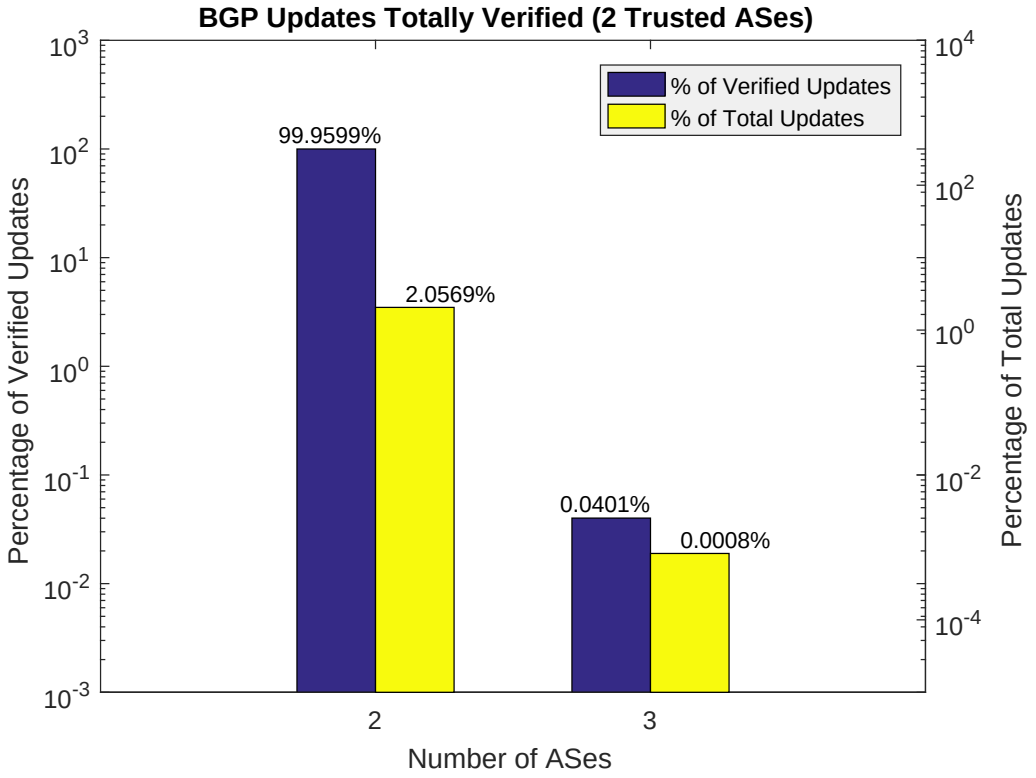


Figure 5.79: Totally Verified BGP Updates - Dataset 10 - 2 Trusted ASes

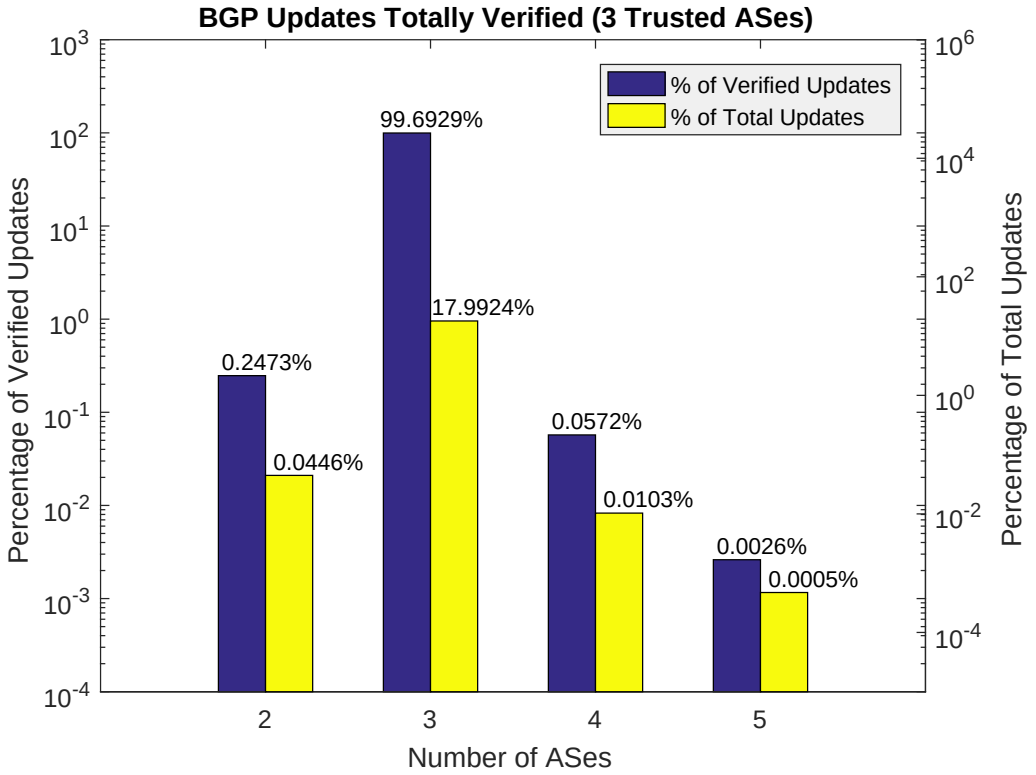


Figure 5.80: Totally Verified BGP Updates - Dataset 10 - 3 Trusted ASes

Bibliography

[1] https://www.nist.gov/image/bgp-youtube-hijackpng, [Accessed 11 January 2022].

[2] https://www.thebalance.com/what-is-blockchain-5088260, [Accessed 11 January 2022].

[3] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the thirteenth EuroSys conference*, pages 1–15, 2018.

[4] Randy Bush and Rob Austein. The resource public key infrastructure (rpki) to router protocol, 2013.

[5] Matt Lepinski and Kotikalapudi Sriram. Bgpsec protocol specification. *RFC8205*, 2017.

[6] Muhammad Shahbaz Sean P. Donovan Brandon Schlinker Nick Feamster Jennifer Rexford Scott Shenker Russ Clark Arpit Gupta, Laurent Vanbever and Ethan Katz-Bassett. Sdx: A software defined internet exchange. In *SIGCOMM'14*, Chicago, USA, 2014.

[7] Arpit Gupta, Robert MacDavid, Rudiger Birkner, Marco Canini, Nick Feamster, Jennifer Rexford, and Laurent Vanbever. An industrial-scale software defined internet exchange point. In *13th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 16)*, pages 1–14, 2016.

[8] PATRICK MCDANIEL TONI FARLEY and KEVIN BUTLER. A survey of bgp security issues and solutions. In *Proceedings of the IEEE*, 2008.

[9] Tony Bates, Ravi Chandra, Dave Katz, and Yakov Rekhter. Multiprotocol extensions for bgp-4. Technical report, RFC 2858, June, 2000.

[10] Anja Feldmann Nikolaos Chatzis Jan Boettger Philipp Richter, Georgios Smaragdakis and Walter Willinger. Peering at peerings: On the role of ixp route servers.

In *Proceedings of the 2014 Conference on Internet Measurement Conference*, NYC, USA, 2014.

[11] Euro-ix.net. 2022. euro-ix. [online] available at: ¡https://www.euro-ix.net/en/¿, [Accessed 9 January 2022].

[12] Xin Hu and Z Morley Mao. Accurate real-time identification of ip prefix hijacking. In *2007 IEEE Symposium on Security and Privacy (SP'07)*, pages 3–17. IEEE, 2007.

[13] Peter Hummon Sharon Goldberg, Michael Schapira and Jennifer Rexford. How secure are secure interdomain routing protocols? In *SIGCOMM'10*, New Delhi, India, 2010.

[14] Philip Hunter. Pakistan youtube block exposes fundamental internet security weakness: Concern that pakistani action affected youtube access elsewhere in world. *Computer Fraud & Security*, 2008(4):10–11, 2008.

[15] Patrick McDaniel, William Aiello, Kevin Butler, and John Ioannidis. Origin authentication in interdomain routing. *Computer Networks*, 50(16):2953–2980, 2006.

[16] Alvaro Retana. Secure origin bgp (sobgp). In *NANOG28 Meeting*, 2003.

[17] Stephen Kent, Charles Lynn, and Karen Seo. Secure border gateway protocol (s-bgp). *IEEE Journal on Selected areas in Communications*, 18(4):582–592, 2000.

[18] Edmund L Wong, Praveen Balasubramanian, Lorenzo Alvisi, Mohamed G Gouda, and Vitaly Shmatikov. Truth in advertising: Lightweight verification of route integrity. In *Proceedings of the twenty-sixth annual ACM symposium on Principles of distributed computing*, pages 147–156, 2007.

[19] Chiara Orsini, Alistair King, Danilo Giordano, Vasileios Giotsas, and Alberto Dainotti. Bgpstream: a software framework for live and historical bgp data analysis. In *Proceedings of the 2016 Internet Measurement Conference*, pages 429–444, 2016.

[20] Johann Schlamp, Ralph Holz, Quentin Jacquemart, Georg Carle, and Ernst W Biersack. Heap: reliable assessment of bgp hijacking attacks. *IEEE Journal on Selected Areas in Communications*, 34(6):1849–1861, 2016.

[21] Mattijs Jonker, Aiko Pras, Alberto Dainotti, and Anna Sperotto. A first joint look at dos attacks and bgp blackholing in the wild. In *Proceedings of the Internet Measurement Conference 2018*, pages 457–463, 2018.

[22] Nico Hinze, Marcin Nawrocki, Mattijs Jonker, Alberto Dainotti, Thomas C Schmidt, and Matthias Wählisch. On the potential of bgp flowspec for ddos mitigation at two

sources: Isp and ixp. In *Proceedings of the ACM SIGCOMM 2018 Conference on Posters and Demos*, pages 57–59, 2018.

[23] Pavlos Sermpezis, Vasileios Kotronis, Petros Gigis, Xenofontas Dimitropoulos, Danilo Cicalese, Alistair King, and Alberto Dainotti. Artemis: Neutralizing bgp hijacking within a minute. *IEEE/ACM Transactions on Networking*, 26(6):2471–2486, 2018.

[24] Christoph Dietzel, Matthias Wichtlhuber, Georgios Smaragdakis, and Anja Feldmann. Stellar: network attack mitigation using advanced blackholing. In *Proceedings of the 14th International Conference on emerging Networking EXperiments and Technologies*, pages 152–164, 2018.

[25] Cecilia Testart, Philipp Richter, Alistair King, Alberto Dainotti, and David Clark. Profiling bgp serial hijackers: capturing persistent misbehavior in the global routing table. In *Proceedings of the Internet Measurement Conference*, pages 420–434, 2019.

[26] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, page 21260, 2008.

[27] Dylan Yaga, Peter Mell, Nik Roby, and Karen Scarfone. Blockchain technology overview. *arXiv preprint arXiv:1906.11078*, 2019.

[28] Julien Polge, Jérémy Robert, and Yves Le Traon. Permissioned blockchain frameworks in the industry: A comparison. *Ict Express*, 7(2):229–233, 2021.

[29] Chris Dannen. *Introducing Ethereum and solidity*, volume 318. Springer, 2017.

[30] Peter Eze, Tochukwu Eziokwu, and Chinedu Okpara. A triplicate smart contract model using blockchain technology. *Circulation in Computer Science–Disruptive Computing, Cyber-Physical Systems (CPS), and Internet of Everything (IoE)*, pages 1–10, 2017.

[31] Richard Gendal Brown, James Carlyle, Ian Grigg, and Mike Hearn. Corda: an introduction. *R3 CEV, August*, 1:15, 2016.

[32] Ajay Kumar Shrestha, Ralph Deters, and Julita Vassileva. User-controlled privacy-preserving user profile data sharing based on blockchain. *arXiv preprint arXiv:1909.05028*, 2019.

[33] Vikram Dhillon, David Metcalf, and Max Hooper. The hyperledger project. In *Blockchain enabled applications*, pages 139–149. Springer, 2017.

[34] Joao Sousa, Alysson Bessani, and Marko Vukolic. A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform. In *2018 48th annual IEEE/IFIP international conference on dependable systems and networks (DSN)*, pages 51–58. IEEE, 2018.

[35] Nicolae Berendea, Hugues Mercier, Emanuel Onica, and Etienne Riviere. Fair and efficient gossip in hyperledger fabric. In *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, pages 190–200. IEEE, 2020.

[36] Faizan Safdar Ali and Alptekin Kupcu. Improving pki, bgp, and dns using blockchain: A systematic review. *arXiv preprint arXiv:2001.00747*, 2020.

[37] Karen Lewison and Francisco Corella. Backing rich credentials with a blockchain pki. *Pomcor. com*, 2016.

[38] Harry A Kalodner, Miles Carlsten, Paul Ellenbogen, Joseph Bonneau, and Arvind Narayanan. An empirical study of namecoin and lessons for decentralized namespace design. In *WEIS*. Citeseer, 2015.

[39] Ryan Shea Muneeb Ali, Jude Nelson and Michael J. Freedman. Blockstack: A global naming and storage system secured by blockchains. In *Proceedings of the 2016 USENIX Annual Technical Conference (USENIX ATC '16)*, Denver, USA, 2016.

[40] Guobiao He, Wei Su, Shuai Gao, Jiarui Yue, and Sajal K Das. Roachain: Securing route origin authorization with blockchain for inter-domain routing. *IEEE Transactions on Network and Service Management*, 2020.

[41] Adiseshu Hari and T.V. Lakshman. The internet blockchain: A distributed, tamper-resistant transaction framework for the internet. In *HotNets-XV*, Atlanta, USA, 2016.

[42] Danny Dolev and Andrew Yao. On the security of public key protocols. *IEEE Transactions on information theory*, 29(2):198–208, 1983.

[43] Shinyoung Cho, Romain Fontugne, Kenjiro Cho, Alberto Dainotti, and Phillipa Gill. Bgp hijacking classification. In *2019 Network Traffic Measurement and Analysis Conference (TMA)*, 2019.

[44] https://en.wikipedia.org/wiki/bgp$_h ijacking, Accessed 12 April 2022.$

[45] Avichai Cohen, Yossi Gilad, Amir Herzberg, and Michael Schapira. Jumpstarting bgp security with path-end validation. In *Proceedings of the 2016 ACM SIGCOMM Conference*, SIGCOMM '16, 2016.

[46] https://bgpupdates.potaroo.net/instability/bgpupd.html, Accessed 12 April 2022.

[47] Wilson S Melo, Alysson Bessani, Nuno Neves, Altair Olivo Santin, and Luiz F Rust C Carmo. Using blockchains to implement distributed measuring systems. *IEEE transactions on instrumentation and measurement*, 68(5):1503–1514, 2019.

[48] https://en.wikipedia.org/wiki/bitcoin$_s calability_p roblem, Accessed 12 April 2022.

[49] Sharon Goldberg. Why is it taking so long to secure internet routing? In *Communications of the ACM*, 2014.

[50] Yury Yanovich, Ivan Ivashchenko, Alex Ostrovsky, Aleksandr Shevchenko, and Aleksei Sidorov. Exonum: Byzantine fault tolerant protocol for blockchains. *bitfury. com*, pages 1–36, 2018.

[51] https://medium.com/@grsind19/summary-of-hyperledger-projects-open-source-enterprise-blockchain-technologies-2386c91e78f3, [Accessed 11 January 2022].

[52] Christian Cachin et al. Architecture of the hyperledger blockchain fabric. In *Workshop on distributed cryptocurrencies and consensus ledgers*, volume 310, pages 1–4. Chicago, IL, 2016.

[53] Ze Wang, Jingqiang Lin, Quanwei Cai, Qiongxiao Wang, Daren Zha, and Jiwu Jing. Blockchain-based certificate transparency and revocation transparency. *IEEE Transactions on Dependable and Secure Computing*, 2020.

[54] Alexander Yakubov, Wazen Shbair, Anders Wallbom, David Sanda, et al. A blockchain-based pki management framework. In *The First IEEE/IFIP International Workshop on Managing and Managed by Blockchain (Man2Block) colocated with IEEE/IFIP NOMS 2018, Tapei, Tawain 23-27 April 2018*, 2018.

[55] I Sentana, Muhammad Ikram, and Mohamed Ali Kaafar. Blockjack: Towards improved prevention of ip prefix hijacking attacks in inter-domain routing via blockchain. *arXiv preprint arXiv:2107.07063*, 2021.

[56] Alfonso de la Rocha Gómez-Arevalillo and Panos Papadimitratos. Blockchain-based public key infrastructure for inter-domain secure routing. In *international workshop on open problems in network security (iNetSec)*, pages 20–38, 2017.

[57] Qianqian Xing, Baosheng Wang, and Xiaofeng Wang. Bgpcoin: Blockchain-based internet number resource authority and bgp security solution. *Symmetry*, 10(9):408, 2018.

[58] Jordi Paillisse, Jan Manrique, Guillem Bonet, Alberto Rodriguez-Natal, Fabio Maino, and Albert Cabellos. Decentralized trust in the inter-domain routing infrastructure. *IEEE access*, 7:166896–166905, 2019.

[59] Emin Gün Sirer Ittay Eyal, Adem Efe Gencer and Robbert van Renesse. Bitcoin-ng: A scalable blockchain protocol. In *Proceedings of the 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI '16)*, Santa Clara, USA, 2016.

[60] https://www.manrs.org/2021/03/a-regional-look-into-bgp-incidents-in-2020/, Accessed 12 April 2022.