# Blockchain for the management of Internet Resources

by

Stefano Angieri

A dissertation submitted by in partial fulfillment of the requirements for the degree of Doctor of Philosophy in

Telematics Engineering

Universidad Carlos III de Madrid

Advisor: Marcelo Bagnulo
Co-Advisor: Alberto García-Martínez

October 2021

*Blockchain for the management of Internet resources*

Prepared by:
Stefano Angieri, Universidad Carlos III de Madrid
contact: sangieri@pa.uc3m.es

Under the advice of:
Marcelo Bagnulo,
Department of Telematics Engineering,Universidad Carlos III de Madrid
Alberto García-Martínez,
Department of Telematics Engineering, Universidad Carlos III de Madrid



*"Chancellor on brink of second bailout of banks"*
*Bitcoin Genesis Block - The Time 3 Jan 2009*

*"With $2.3T Injection, Fed's Plan Far Exceeds 2008 Rescue"*
*Bitcoin Block 629999, Third Halving - NYTimes 09 Apr 2020*

*"Dicette o pappice vicino a' noce ramm' o tiemp' ca te spertose".*
*"Said the maggot to the tree, Lend me time and I'll pierce thee"*
*Proverbio Napoletano*

# Acknowledgements

The realisation of this thesis is the outcome of an intense journey, known as PhD. As this journey has come to an end, I want to use this moment and this page to thank anyone who has been part of it.

My exceptional gratitude and sincere appreciation goes to my advisors Marcelo Bagnulo and Alberto García-Martínez who both flanked me throughout the development of this thesis. Working with you has been a meaningful experience for my professional growth. Your collaboration, support and guidance have been fundamental for reaching all the achievements this thesis builds upon, and to let me develop the scientific thinking I have today. It has been a lucky break to meet and work with two amazing scientists and people like you are.

A Mamma, Papà e mio Fratello per essere stati SEMPRE presenti. Il vostro sostengo ed incoraggiamento sono stati incodizionati e fondamentali in tutto quel percorso che, dal fasciatoio, oggi mi porta ad essere qui. GRAZIE.

To all my colleagues and friends from UC3M and IMDEA, and to all the amazing people I had the luck to share this journey with. When you feel like you are in a expanded family, everything in life become easier and achievable.

A Luigi e Boris e alla fortuna di avervi accanto in questa vida Madrileña.

A ti Palomita, por haber llegado a mi vida durante el atardecer de este viaje.

# Published and Submitted Content

This thesis is based on the following published and submitted papers:

**[1]** **Stefano Angieri**, Alberto García-Martínez, Bingyang Liu, Zhiwei Yan, Chuang Wang and Marcelo Bagnulo, "A Distributed Autonomous Organization for Internet Address Management" in *IEEE Transactions on Engineering Management, vol. 67, no. 4, pp. 1459–1475, 2020.*

- This work is partially included and its content is reported in Chapter 1, 2, 3, 4.
- The author's role in this work is focused on the study of the state of the art, collaboration in the design and as well implementation and experimentation of the concepts proposed in the paper.

**[2]** Alberto García-Martínez, **Stefano Angieri**, Bingyang Liu, Fei Yang and Marcelo Bagnulo, "Design and Implementation of InBlock—A Distributed IP Address Registration System", in *IEEE Systems Journal, vol. 15, no. 3, pp. 3528–3539, 2021.*

- This work is partially included and its content is reported in Chapter 1, 2, 3, 4.
- The author's role in this work is focused on the study of the state of the art, collaboration in the design and as well implementation and experimentation of the concepts proposed in the paper.

**[3]** **Stefano Angieri**, Marcelo Bagnulo, Alberto García-Martínez, Bingyang Liu and XinPeng Wei, "InBlock4: Blockchain-based Route Origin Validation", in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2020, pp. 291–296* .

- This work is partially included and its content is reported in Chapter 1, 2, 3, 4.
- The author's role in this work is focused on the study of the state of the art, collaboration in the design and as well implementation and experimentation of the concepts proposed in the paper.

[4] **Stefano Angieri**, Marcelo Bagnulo, Alberto García-Martínez,"Internet Routing Blockchain: an Hyperledger Fabric consortium blockchain for Internet Routing Registries". This work has been submitted for publication.

- This work is partially included and its content is reported in Chapter 1, 2, 3, 5.
- The author's role in this work is focused on the study of the state of the art, collaboration in the design and as well implementation and experimentation of the concepts proposed in the paper.

[5] Marcelo Bagnulo, Alberto García-Martínez, **Stefano Angieri**, Andra Lutu and Jinze Yang, "Practicable route leak detection and prevention with ASIRIA". This work has been submitted for publication.

- This work is partially included and its content is reported in Chapter 1, 3, 5.
- The author's role in this work is focused on the study of the state of the art, collaboration in the design and as well collaboration in implementation and experimentation of the concepts proposed in the paper.

# Abstract

3 January 2009, the first version of the Bitcoin software was released. That date has been the really first time that blockchain technology has been revealed to the world. A blockchain is an immutable distributed ledger that records validated transactions permanently employing cryptography and a distributed consensus algorithms. All the information is stored and validated by all the nodes composing the peer-to-peer network without the need of a trusted third party. Bitcoin is the first application of this technology and it allows e-payments to be easily performed between two parties with no need of intermediaries. Since then the hype about the technology has led several advances, exceeding the pure financial sphere and attempting to solve disparate problems, rather than digital payments only. As example, Ethereum with its introduction of smart contracts, has been the first application of the so called 2.0 generation of blockchain technology. A smart contract is a self-executing program, stored in the blockchain, with the terms of the agreement between interested parties hard-coded in the contract definition. Smart contract allow trusted transactions and agreements to be performed between distrust parties without the need for any intermediary or external enforcement mechanism. Therefore, with the introduction of smart contracts, the blockchain paradigm can be extended to the automation of complex resource manipulation and transference procedures in a transparent and trustable manner, by means of the specification of these new types of contracts.

Blockchain and smart contracts technology is being adopted in a vast number of scenarios, including finances, Internet of Things, health care, energy, education, and more. As with any novel technology, there are many open questions about its usage and potential. For this reason, we believe that practical experimentation is in order to have hands-on experiences. So we present this research work on the application of the blockchain technology for the management of internet resources.

The first study we present in this thesis is the design of a blockchain based application to build a decentralised IP address registry. Empowering the blockchain technology we propose to change by design the centralized structure of the current system to manage the global

pool of IP addresses and the centralized and hierarchical model that is implemented in the Resource Public Key Infrastructure (RPKI) that makes lower layers in the hierarchy susceptible to errors and abuses from entities placed in higher layers. Hence we present the design of InBlock, a Distributed Autonomous Organization (DAO) that provides decentralized management of IP addresses. The InBlock automates the process of assigning Internet resources to the user complying with the "law" written in its smart contracts. InBlock also fulfills the same objectives as the current IP address allocation organizations, i.e., uniqueness, fairness, conservation, aggregation, registration and minimized overhead. InBlock is implemented as a set of blockchain's smart contracts in Ethereum and it implements all the functions needed for the management of a global pool of addresses without any human intervention. Moreover InBlock embeds an alternative trust model to the hierarchical one currently implemented by the RPKI.

In this thesis we present two Proof of Concept (PoC) implementation of InBlock: InBlock6 and InBlock4. InBlock6 implements the InBlock design and is centred on the management of the IPv6 address space, that compared to the IPv4 address space, has way more free resources that can be assigned. InBlock4 inherits its functionalities but for the IPv4 address space and provides an alternative framework to register living resources (e.g. already assigned resources) into the blockchain to enable the decentralised route origin validation. We present the implementation and evaluation of both the PoC for the Ethereum blockchain and we quantify their performance.

The second study we present in this thesis is on the design and a PoC implementation of the Internet Routing Blockchain (IRB), an implementation of the Internet Routing Registry (IRR) functionality within Hyperledger Fabric (HF). The IRR is a distributed routing database that provides a mechanism for validating the contents of Border Gateway Protocol (BGP) announcement messages and mapping an origin Autonomous System (AS) number to a list of networks [6]. The IRB relies on a permissioned blockchain technology that is inherently distributed, allows to preserve the decentralised nature of the IRR, overcomes the centralized governance model limitation of current used mechanism employing a consortium based model, provides consistency and information stall-ness prevention and offers a simple declaration syntax for the policy definition of ASes relationship.

As the final contribution of this thesis we present a study about route leaks prevention and the utilization of the information stored in the IRR. A route leak is defined as the propagation of a route beyond its intended scope. Those events have historically caused a consistent number of incidents resulting in Internet disconnections and disservices that generates money loss. In this research work we present the design and the performance evaluation of Autonomous System Internet Registry Inference for path Authorization (ASIRIA),

a mechanism for detecting leaked routes and leakage events that uses AS relationship information inferred from the IRR.

# Table of Contents

# List of Tables

# List of Figures

# List of Acronyms

**RPKI** Resource Public Key Infrastructure

**AS** Autonomous System

**ASN** Autonomous System Number

**RIS** Routing Information System

**CPV** Customer-Provider Verification

**ASPA** Autonomous System Provider Authorization

**ASIRIA** Autonomous System Internet Registry Inference for path Authorization

**BGP** Border Gateway Protocol

**BGPsec** Border Gateway Protocol Security

**BFT** Byzantine Fault Tolerance

**CA** Certificate Authority

**C2P** Customer to Provider

**DAO** Distributed Autonomous Organization

**DARPA** Defence Advanced Research Projects Agency

**DNS** Domain Name System

**EOA** Externally Owned Account

**EVM** Ethereum Virtual Machine

**GDP** Gross Domestic Product

**HF** Hyperledger Fabric

**IANA** Internet Assigned Number Authority

**IAB** Internet Architecture Board

**ICANN** Internet Corporation for Assigned Names and Numbers

**IETF** Internet Engineering Task Force

**IRB** Internet Routing Blockchain

**IRR** Internet Routing Registry

**ISOC** Internet Society

**FRR** FRRouting

**ISP** Internet Service Provider

**NFT** Non-Fungible token

**LIR** Local Internet Registry

**NIR** National Internet Registry

**NMC** NameCoin Token

**NRO** Number Resource Organization

**MANRS** Mutually Agreed Norms for Routing Security

**NTI** Non Transit Indicator

**NTIA** National Telecommunications and Information Administration

**OTC** Only to Customer

**P2C** Provider to Customer

**P2P** Peer to Peer

**PA** Provider Aggregatable

**PI** Provider Independent

**PoC** Proof of Concept

**PoS** Proof of Stake

**PoW** Proof of Work

**RIR** Regional Internet Registry

**ROA** Route Origin Authorisation

**ROV** Route Origin Validation

**RPKI-RTR** RPKI to Router protocol

**RPSL** Routing Policy Specification Language

**TABL** Transferable Block Lease

**URI** Uniform Resource Identifier

**ASIRIA LS** ASIRIA Local Server

**CAIDA** Cooperative Association for Internet Data Analysis

*"The measure of greatness in a scientific idea*
*is the extent to which it stimulates thought*
*and opens up new lines of research"*

Paul Dirac

# 1 Introduction

Since the revelation of Bitcoin [7] in 2009, blockchain technology has drawn the world wide attention. Blockchains, and in particular, cryptocurrencies, one of the several blockchain-based applications, have already attracted the consideration of media, business enterprises and governments [8]. The financial interest of this technology is beyond question, with a market Cap of US$ 2.6 trillion for the aggregate of the 5,840 (and rising) different cryptocurrencies [9, 10]. However, blockchain appeal exceeds the financial sphere. Blockchains, as a new general purpose technology yet in an early stage of development, exhibit a potential to disrupt everyday life comparable to computers or Internet itself. A wide adoption of blockchain can bring a higher degree of automation, the progressive elimination of intermediaries, an easier and faster money circulation. For example, it has been estimated that Blockchain can led financial institutions to save $20 billion per year in crossborder payment costs, settlement and regulatory [11]. In addition, blockchain come with the promise of a much yearned transparency in the relationship with companies and institutions [12].

Blockchain technology opens new innovation opportunities. One of the most disruptive prospect emerges from the combination of the code-ification of law paradigm with blockchain smart contracts. This alliance can empower the ex-ante enforcement of technical rules, despite the difficulty and the cost of transposing legal specifications, written in natural language and so inherently ambiguous, into technical rules based on mathematical models and formal algorithms [13]. In this vein, the application of the blockchain technology to the management of the Internet resources result naturally from the convenience to provide a fair and predictable, transparent, automatic and efficient framework. Accordingly, the main motivations for this research work is the need of study and experimentation of the potential application of the blockchain technology for the management of the internet resources.

We next present the main challenges we have faced in this research work, then its main contribution and the thesis outlines.

## 1.1. Challenges

- *Limitations of the hierarchical structure for the Internet resource management*

The current structure for the management of the global pool of IP addresses results in the jurisdiction of the countries where Internet Corporation for Assigned Names and Numbers (ICANN) and the Regional Internet Registries (RIRs) are actually located overflowing into all the other countries served. As ICANN and the RIRs are private organizations, they must operate under the legal framework of the countries where they are hosted. Yet, they manage the IP addresses for a bigger set of countries. As a consequence, for the majority of the countries in the world, the management of a critical Internet resource such as IP addresses obey the legal framework of a foreign country. Thereby, for most countries any legal action involving IP address management has to be settled in a foreign court of law, making these resources *de facto* subject to laws of a foreign state, as observed in [14].

- *RPKI management, potential errors and abuses*

In the past few decades, the Internet has become of utmost importance in the critical infrastructure for most countries in the world. Therefore the need to guarantee its availability has led to the design, deployment and adoption of new security tools, such as the Resource Public Key Infrastructure (RPKI) and Border Gateway Protocol Security (BGPsec). These tools have been developed in order to grant cryptographic guarantees that whoever is claiming to have an Internet addressing resource is actually the legitimate holder of the resource in conformity with the defined allocation rules. Thereby these security tool provide prevention against prefix hijacking attacks and other vulnerabilities. Thus, these mechanisms intrinsically offer to the entities up in the hierarchy of the allocation system (i.e., RIRs, National Internet Registry (NIRs) and Local Internet Registry (LIRs) ) the capacity to actually enforce the allocations in real time. In particular, they allow entities up in the allocation hierarchy to arbitrarily override an existing IP allocation [15]. Consequently with the adoption of the these cryptographic techniques, the Internet Registries will have the capacity to invalidate allocation and disconnecting whole network from the internet at will and if dictated by their governing bodies. As a matter of fact this situation has raised several concerns and it may be one of the reasons behind the slow adoption of these technologies. We note that the attacks they are designed to prevent are very concrete and dangerous, so security measures to protect the Internet are indeed needed.

- *Route Leaks*

Over the last yeas, the internet community effort in securing the Border Gateway Protocol (BGP), has lead to the definition of several BGPsec mechanisms. While these mechanisms protect BGP against several types of common attacks (e.g., prefix hijacks), they fail to prevent another very common threat known as *route leaks*. Route leak events have historically caused a large number of wide-scale disruptions on the Internet. A route leak is defined as the propagation of a route beyond its intended scope [16]. Leaks are particularly hard to detect and prevent because they most frequently involve routes with legitimate origin propagated through legitimate paths that are propagated beyond their legitimate scope.

- *Decentralised management of the information stored in the IRR*

The Internet Routing Registry (IRR) is a distributed routing database that provides a mechanism for validating the contents of BGP announcement messages and mapping an origin Autonomous System (AS) number to a list of networks. The data stored in the IRRs contains information on the business relationship between the ASes which can be used to perform route leaks prevention, as described in Autonomous System Internet Registry Inference for path Authorization (ASIRIA) [5]. The IRR counts with 25 different repositories maintained by different entities. Due to its distributed nature, the quality of the information available can suffer from inconsistency across different IRRs and information stall-ness. Besides, the complexity of the Routing Policy Specification Language (RPSL) syntax, the routing policy definition language, limits the precision of relationship inference algorithms.

## 1.2. Contributions

This thesis investigates the potential of the application of blockchain technology for the management of internet resources. The main contribution of this doctoral thesis have been publish in 5 publications. Contributions and publications can be mapped as follows:

**[1] Stefano Angieri**, Alberto García-Martínez, Bingyang Liu, Zhiwei Yan, Chuang Wang and Marcelo Bagnulo, "A Distributed Autonomous Organization for Internet Address Management" in *IEEE Transactions on Engineering Management, vol. 67, no. 4, pp. 1459–1475, 2020.*

**[2]** Alberto García-Martínez, **Stefano Angieri**, Bingyang Liu, Fei Yang and Marcelo Bagnulo, "Design and Implementation of InBlock—A Distributed IP Address

Registration System", in *IEEE Systems Journal, vol. 15, no. 3, pp. 3528–3539, 2021.*

- *Contribution : InBlock Design*

The first contribution of this thesis is the design of the InBlock, a system to securely manage a global pool of IPv6/IPv4 addresses in a fully de-centralized manner. By relying on blockchain technology, the proposed InBlock design provides a distributed, automatic, irrevocable, tamper-free, publicly accessible, privacy-preserving resource allocation mechanism for the Internet. At the same time, the proposed design of the InBlock complies with the goals stated for the current Internet address assignment mechanism, i.e., uniqueness, fairness, conservation, aggregation, registration and minimized overhead [17]. We illustrate the proposed design by describing the overall operation of the proposed InBlock solution, including the different roles involved and the functions performed by each of them.

- *Contribution : InBlock6 Proof-Of-Concept*

The second contribution of this thesis is a Proof of Concept (PoC) implementation of InBlock6. In this first PoC we focus our attention on the IPv6 address space as it has a large remaining pool of unassigned addresses compared to the IPv4 address space. InBlock6 is implemented as a set of smart contracts in Ethereum [18]. We describe the different *transactions* and *calls* that are implemented to manage the IP address pool. The InBlock6 implementation is open source and publicly available.

- *Contribution : InBlock6 experimental validation*

The third contribution of this thesis is the experimental validation of the proposed InBlock design and implementation. We deployed our InBlock6 implementation both in a private testbed and the public blockchain and assessed its functionality and performance. We systematically test all the *transactions* and *calls* implemented. We observe that InBlock6 is able to perform the IPv6 registry functions in an efficient manner, significantly reducing the operational costs compared to a traditional registry and also reducing in orders of magnitude the time required to perform an allocation. The scripts used to test the implementation are also publicly available to enable the reproducibility of the reported results.

**[3] Stefano Angieri**, Marcelo Bagnulo, Alberto García-Martínez, Bingyang Liu and XinPeng Wei, "InBlock4: Blockchain-based Route Origin Validation", in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2020, pp. 291–296 .*

- *Contribution : InBlock4 Proof-Of-Concept*

The fourth contribution of this thesis is a PoC implementation of InBlock4. InBlock4 extends InBlock6 inheriting its functionalities but for the IPv4 address space, and providing an alternative framework to register living resources (e.g., already assigned resources) into the blockchain to enable the decentralised route origin validation. InBlock4 is implemented as a set of smart contracts in Ethereum. We describe the different *transactions* and *calls* that are implemented to manage the IP address pool. The InBlock4 implementation is open source and publicly available.

- *Contribution : InBlock4 experimental validation*

The fifth contribution of this thesis is the experimental validation of the proposed InBlock4 design and implementation. As for InBlock6, we deployed our InBlock4 implementation both in a private testebed and the public blockchain and assessed its functionality and performance. We systematically test all the *transactions* and *calls* implemented.

[4] **Stefano Angieri**, Marcelo Bagnulo, Alberto García-Martínez, "Internet Routing Blockchain: an Hyperledger Fabric consortium blockchain for Internet Routing Registries". This work has been submitted for publication.

- *Contribution : IRB Design*

The sixth contribution of this thesis is the design of the Internet Routing Blockchain (IRB), a Hyperledger Fabric (HF) based permissioned system to securely manage the information stored in the IRR in a fully de-centralized manner.

- *Contribution : IRB Proof-Of-Concept*

The seventh contribution of this thesis is a PoC implementation of the IRB. The IRB implementation is open source and publicly available.

[5] Marcelo Bagnulo, Alberto García-Martínez, **Stefano Angieri**, Andra Lutu and Jinze Yang, "Practicable route leak detection and prevention with ASIRIA". This work has been submitted for publication.

- *Contribution : ASIRIA Design*

As the eighth contribution of this thesis we introduce ASIRIA, a novel mechanism to provide protection against route leaks based on drop-lists. We propose a shift in the

paradigm on the use of IRR information to prevent route leaks, from the current accept-list one to an alternative one based on drop-lists. ASIRIA infers the type of relationships between ASes from the information available in the IRR. Using the information about ASes relationships, ASIRIA identifies and avoid invalid routes i.e. leaks. Invalid paths, are detected by analyzing the relationships between consecutive ASes present in the AS_PATH attribute of a BGP route.

- *Contribution : ASIRIA performance evaluation*

The ninth contribution of this thesis is a performance evaluation of ASIRIA. In this quantitative analysis we measure how many paths marked as leaks by our mechanism do not correspond to real leaks. We determine an upper bound for this value. In addition, we compute the sensitivity of the ASIRIA alarm system for detecting leakage events, i.e. an event that generates a set of leaks. By relying on existing information, ASIRIA provides immediate benefits to early adopters. In particular, we analyse over 300 ASes and we show that 90% of analyzed ASes can detect over 99% of the leakage events generated by a customer or a peer solely using currently available information.

## 1.3. Outline

This thesis is structured in six chapters. In chapter one we have presented an introduction to this research work pointing out its main challenges and contribution. In chapter two we gave insights about the state of the art over the blockchain technology, Internet resource management and the RPKI. In chapter three we show the current research work regarding the application of the blockchain technology for the management of Internet resources and about the route leak prevention. In chapter four we first discuss in detail the challenges faced by our solution, then we present the InBlock general design and finally the two PoC, namely InBlock6 and InBlock4, with their implementation aspects and evaluation. In chapter five we first discuss in detail the research challenges, then we present the IRB PoC, a private blockchain consortium for the distributed management of the information stored in IRRs. Later on we illustrate the design and the evaluation of ASIRIA, a novel mechanism to provide protection against route leaks. Finally conclusions are presented in chapter six.

# 2 State of the art

In this chapter we first introduce the blockchain technology, with emphasis on two of its incarnations, Ethereum and Hyperledger Fabric (HF), as they are the platforms of choice for developing the InBlock and the Internet Routing Blockchain (IRB) solution.

Later on in section 2.2 we describe the current IP address allocation system and the entities involved. In order to understand which are the measures that current key entities can apply to restrict the ability of third parties to communicate, we sketch the basic notions of the Internet routing system, and the cryptographic security standards proposed. Moreover we give insight about the Border Gateway Protocol (BGP) security.

## 2.1. Background on Blockchains

### 2.1.1. Blockchain Technology

The blockchain or distributed ledger technology is a smart combination of three leading technologies: *cryptography*, a *peer-to-peer network containing a shared ledger* and a *distributed consensus mechanisms* to store the transactions and records of the network.

A blockchain is an immutable distributed ledger that records validated transactions permanently without the need of a trusted third party. The blockchain is a distributed ledger because all the information is stored in all the nodes composing the blockchain peer-to-peer network[1]. The blockchain is composed of an append-only list of blocks, securely linked between each other through cryptography [19]. Each individual participating in the network is identified by a couple of keys, a *private key* and a *public key*, which are used to produce a secure digital identity reference for authorizing and controlling transactions.

[1]This is different than other distributed databases, where different parts of the database are stored in different nodes with a limited level of replication just to achieve redundancy and performance benefits

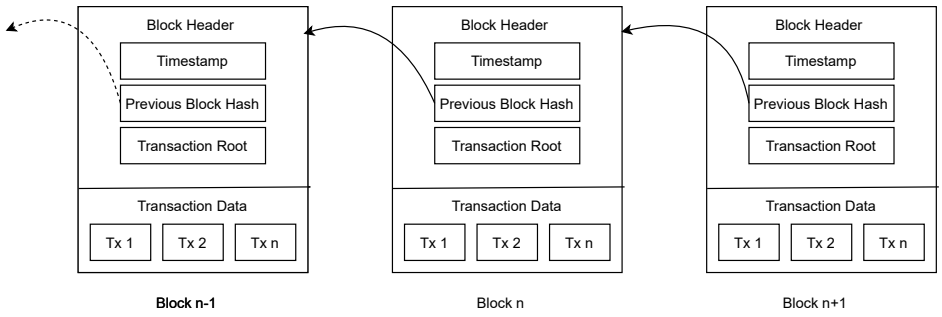


Figure 2.1: Blockchain structure

As shown in figure 2.4, every block contains a hash pointer to a parent block, a timestamp and transactions data. The addition of new valid blocks is determined through a distributed consensus mechanism. The result of this process is considered as the network definitive truth and it represent the agreement reached by more than thousands of nodes on the blocks added to the blockchain. The most popular consensus mechanism is Proof of Work (PoW) [20]. In PoW, nodes try to solve a complex mathematical problem in order to gain the right to append a block to the existent chain (and make some profit). New block signers are chosen through a *mining* race. Every time a block is added, a new mining race starts and every miner tries to find the solution to gain the next mining block contest and receiving the related fee. Since every miner is working on the same problem, once the challenge is solved by someone, the computational power spent by the other miners on the same problem is wasted. As a consequence the PoW consensus mechanism is considered as the most expensive in terms of energy consumption.

The hash structure of the blockchain makes computationally unfeasible to alter the data of one block without the manipulation of all subsequent blocks. Tampering the ledger then requires both the collusion of the majority of the network and an enormous amount of computational power to rebuild the chain from the replaced block. This is the sense in which we interpret the immutability of the blockchain. Finally, we stress that only valid transactions are included in the blocks forming the blockchain. To determine if a transaction is valid, all nodes participating comply with the same block validation rules. As an example of these rules, a valid transaction must be signed with the private key of the originator.

The blockchain paradigm can be extended to the automation of complex resource manipulation and transference procedures in a transparent and trustable manner, by means of the specification of *smart contracts*. A smart contract [21,22] is a deterministic programme that executes business logic in response to event and that is stored and run in the blockchain. Once deployed, the blockchain nodes will execute the smart contract whenever a monetary

transaction to the contract account triggers its execution. As a consequence of their flexibility, Blockchain and smart contracts are being adopted in a vast number of scenarios, including finances, Internet of Things, health care, energy, education and more [23]. Blockchains can be indeed public, private or permissioned. While in public blockchain everyone has the right to join, participate in the network and as well the ability to read information from the distributed ledger itself, in private or permissioned blockchains users must be authorized by the majority or by a predefined subset of the peers to accomplish a set of pluggable operations.

We now discuss the technological aspects related to the specific framework we have used for the solution that we propose in this research work, Ethereum and HF.

### 2.1.2. Ethereum

For the implementation of the IP address registry we have chosen Ethereum. We next justify the selection of Ethereum for this purpose and then we cursorily analyse if Ethereum can provide the latency, throughput and cost required by InBlock.

Ethereum [18] is a public blockchain platform created to facilitate the development of smart contracts. Ethereum has a built-in Turing-complete programming language, known as solidity, that allow developers to easily write smart contracts. Smart contracts are compiled to opcodes that are executed in the Ethereum Virtual Machine (EVM). The EVM is a safe environment, where Ethereum opcodes get executed, that guarantees to the network portability and robustness.

Every operation in the network is triggered by transactions between accounts, either Externally Owned Accounts (EOAs), owned and controlled by users, or Contract Accounts, associated to a Smart contract which code and state are stored with the account itself. Being a public blockchain, any party can create one or more EOAs and run (or deploy) a smart contract in Ethereum.

Ethereum has implemented a PoW based consensus mechanism and currently is moving to Proof of Stake (PoS) consensus mechanism. The reader is referred to [24] for detail regarding the consensus mechanism transition.

Miners are rewarded in Ether, Ξ, the Ethereum cryptocurrency, for the storage and processing power they contribute to. Ethereum users that want to run a smart contract or to use one, issue a transaction in the Ethereum network which includes a transaction fee payable to the miners. The value (in Ether) of the transaction fee is set by the user issuing the transaction and should reflect the execution and storage cost of an operation, as well as the priority that the user wants to get from the blockchain miners. Higher transaction fees imply that the transaction will be processed earlier by the miners. The fee of a transaction is represented in Ethereum through the concept of *gas*. A fixed amount of gas is assigned

to each operation and each transaction sets its own *gas price*. The amount of Ether that the issuer of the transaction transfers to the miner is the amount of gas required by the computational cost of the transaction execution multiplied by the gas price offered. Also, every transaction defines the gas limit field to set the maximum amount of gas that the transaction may consume. If during the runtime of the code associated to a transaction the gas used exceeds the gas-limit defined for the transaction, the processing is stopped with an *out-of-gas* error. Finally, Ethereum defines a global upper bound to the gas limit value for any single transaction. Transaction confirmation times are estimated around 10-15 seconds depending on storage needs, code complexity and bandwidth usage.

Ethereum Smart Contracts can also include *calls*, code that does not generate blockchain annotations and does not incur in a transaction cost. A call operates over the blockchain data to retrieve information, perform validations, etc. Calls can be used as a mean to the smart contract designer to provide a canonical way of retrieving, elaborating or validating blockchain information. The execution of this code provides a higher compliance with the intentions of the smart contract designer, as opposed to an equivalent specification in pseudo-code, or a third-party implementation. Ethereum also imposes gas restrictions on the execution of calls. Ethereum enables the deployment of a Distributed Autonomous Organization (DAO) [25], an organization that is fully implemented in the form of one or more smart contracts without any human involved in the daily operation of the organization. In particular, the bylaws of the organization are embedded into the code of the smart contracts. DAO's financial transaction records and program rules are maintained on the blockchain. Then, every node validating the blockchain will execute the deterministic code of the contract, written in the blockchain itself, reaching the same final state.

The reader is referred to [18] and [26] for further information regarding blockchains in general and Ethereum technology in particular.

#### 2.1.2.1. The Ethereum choice

In this section, Ethereum restrictions are analysed, to show that price, latency or throughput of blockchain operations are appropriate for address allocation needs.

We design InBlock, presented in chapter 4, as a set of smart contracts on top of Ethereum. Previous proposals [27–29] propose to create a new blockchain to store information regarding IP address allocation (see section 3 for further details about these proposals). In InBlock, instead of creating a new blockchain, we propose to use an existing one. We believe this approach provides three important benefits.

First, it provides a clean architecture with layered design that separates the blockchain from the registry service. This allows the evolution of the blockchain without affecting the

registry service. For example, there is an ongoing debate regarding whether PoW approaches are sustainable (due to their expensive cost in terms of power consumption) and whether PoS provides a more sustainable alternative. By laying the InBlock on top of an existing (and evolving) blockchain, we make InBlock agnostic to the consensus mechanism. In particular, Ethereum currently uses PoW (which is the proven technology) and it is experimenting with PoS [30]. Once/if PoS is proven and stable, Ethereum will migrate to PoS and InBlock will benefit from this technological advance without any impact in the management of addresses.

Second, by using an existing blockchain, we can rapidly develop and deploy InBlock. For example, Ethereum is already available and working. By using Ethereum, we reduce the development time, as we only need to focus in the implementation of the registry service. All blockchain code evolution and testing[2] are taken care of by the Ethereum community.

Third, using an existing blockchain provides secure bootstrapping, i.e., a secure blockchain from the start of InBlock. Blockchain security heavily depends on the consensus mechanism used (e.g., PoW, PoS). The level of security of the consensus mechanism frequently depends on the level of adoption of the blockchain. A blockchain using PoW is as strong as the hashing power used to mine blocks [20]. In a new blockchain, it is likely that there will be little hashing power as there is little economical gain from mining it. If PoS is used instead, the level of security depends on the number of stakeholders, the concentration of stake and also how much actual value is there in the blockchain [20, 32]. In a new PoS blockchain, it is likely that there will significant concentration of stake and a reduced number of stakeholders control the blockchain. In the case of InBlock, the IPv6 addresses managed by the InBlock registry are valuable per se, prior to the existence of the blockchain. So, if the blockchain provides little security in its early days, there is a risk of an attack directed towards the illegitimate acquisition of allocations. The InBlock must then provide strong security from the bootstrap to be viable and using Ethereum achieves this goal.

Due to the facilities provided to develop smart contracts, and its relative maturity, Ethereum is our blockchain of choice for the InBlock experiment. While the use of Ethereum seems to provide many advantages when building InBlock, we need to verify that Ethereum is also a good fit in terms of performance and cost. We next analyse different relevant parameters:

- **Timescale**. There are roughly three delays involved in an Ethereum transaction, namely, the time it takes for a miner to include a transaction in a block that it is mining, the time it takes to mine the block containing the transaction and the time it takes for the block containing the transaction to be confirmed. Regarding the first delay, miners receive many transactions that are candidates to be included in the next block

[2]Notable bugs have been identified in different blockchains in the past [31].

to be mined. Miners determine which transactions may be included in the current block according to the transaction fee offered(transactions that are willing to pay more get in the blockchain earlier). According to current prices [33], if the InBlock transaction is willing to pay 2 US$ per allocation, the delay to be included in the next block is less than 2 minutes. Regarding the second type of delay, Ethereum publishes a new block every 17 seconds. Finally, assuming that InBlock will wait for 12 new blocks to confirm the transactions containing the allocations [34], this means that it will take about 3 more minutes to confirm the allocation transaction. So, the total delay for an InBlock transaction to be published and confirmed in Ethereum will be in the order of 5 minutes. Currently it takes days to grant new allocations, so the timescale provided by Ethereum is much shorter than the one provided by the current allocation system.

- **Throughput**. Ethereum currently has a mean throughput of 20 transactions per second. In order to estimate the maximum throughput that could be required by InBlock, we compute the number of transaction that it would require renewing yearly all the current IPv4 and IPv6 allocations, plus the transaction resulting from the new IPv4 and IPv6 allocation done every year. This results in 58,700 transactions per year, which results in 0.0019 transactions per second, which is much less than the transaction throughput currently supported by Ethereum.

- **Cost**. In order to run in Ethereum, InBlock needs to pay to the Ethereum network in the form of a transaction fee. Executing InBlock implies the following expenses. First, the deployment of the InBlock code in the Ethereum network requires a transaction fee. We estimate roughly between US$ 15 and US$ 65 for this (depending on the gas cost selected). Once deployed, we estimate in around US$ 0.5 the costs of the transactions required to assign a block. There are other transactions required to run InBlock that will incur in additional costs (e.g., the cost of an oracle converting US$ to Ether is about 1 US$, setting a ROA costs around 0.1 US$). Note that the cost of a transaction fee is likely to increase as Ethereum becomes popular. If InBlock defines a fixed transaction fee according to current values, it is possible that this value will become outdated and that InBlock transaction become unattractive for miners to include them in new blocks they are mining. Because of this, the transaction fee used by the InBlock must be updated to reflect the values expected by the miners at every time. This can be done in Ethereum implementing an "oracle" [35] that provides the InBlock with updated values to use for transaction fees.

### 2.1.3. Hyperledger Fabric

In this section we describe HF, the framework that has been used for the IRB 5.2 solution.

HF is an open source enterprise-grade permissioned distributed ledger technology platform which provides an open and modular architecture and plug-and-play components to accommodate a wide range of use cases. Considered as a de facto standard for enterprise blockchain platforms [36], HF is used for private and consortium blockchain. In a permissioned blockchain, the consortium, i.e., the set of organization maintaining the network, have the control over the consortium admission management while every organizations have the control over its client's membership management. Using HF we can build a consortium blockchain where every entity or organizations interested in its maintenance is allowed to join. An HF blockchain is composed of a number of network components, run by independent

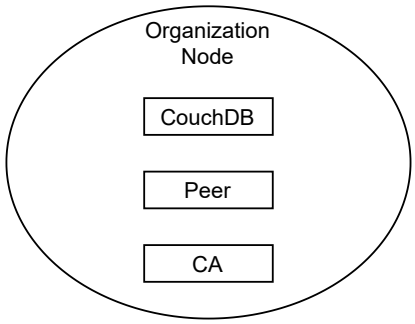


Figure 2.2: Organization Node

*organizations*, and of an ordering service node. As shown in figure 2.2, all the organizations willing to join the network are at least required to deploy the following HF components: a Certificate Authority (CA) and one peer. The CA is in charge of identity registration, enrolment certificate (X.509 certificates are used for the authorization and authentication) issuance, renewal and revocation [37]. Every organization has its own CA by which it manages the identities of its clients. The Peer is the main network component which participates in the consensus, hosts ledgers and smart contracts that in HF contest are known as chaincodes [38]. An endorsement policy has to be specified in every chaincode definition, and it defines the list of peers, participating in the network, that must execute the chaincode to endorse the execution results in order for the transaction to be considered valid. The Couchdb is a nosql alternate state database that allows to model data on the ledger as JSON and offers "rich queries" against data values rather than being restricted to queries to the keys [39]. CouchDB allows data retrieval via simple HTTP requests.

We next give an overview of the transactional mechanism employed by HF. In this context we refer to channel as a blockchain network.

#### 2.1.3.1. Transaction life cycle, validation and endorsement

The transactional mechanics that take place during an asset exchange is a standard process [40]. The *Client* submits the transaction proposal to a set of *Endorsing Peers*, collects the list of transaction responses and once it has received enough responses to satisfy the required endorsement policy, he can broadcast the transaction message to the *Ordering Service*. The transaction message contains info regarding read/write sets generated by the previous process, *Endorsing Peers* signatures and the Channel ID. The *Ordering Service* receives transaction from all channels in the network, orders them chronologically by channel, and creates blocks of transactions per channel. At this stage, ordering nodes deliver new blocks to all peers on their channel. Once a peer receives a new block he validates all the transactions within the block to ensure endorsement policy is fulfilled and to ensure that there have been no changes to ledger state for read set variables since the read set was generated by the transaction execution. Transactions in the block are tagged as being valid or invalid. Then each peer appends the block to his channel's chain, and for each valid transaction the write sets are committed to current state database.

#### 2.1.3.2. The Hyperledger Fabric choice

In this thesis we have simulated a set up of a consortium blockchain maintained by Regional Internet Registries (RIRs). The reason behind that choice are revealed in section 5.3.1. We next discuss the benefits of using a permissioned blockchain for the implementation of the IRB Proof of Concept (PoC)

The usage of a consortium blockchain overcomes several limitation that may come using a public blockchain network [41] regarding the exposure of sensitive information, related to resources, and the resource control. It may seem contradictory to the use of a public blockchain with controlling the exposure of the information, but a consortium permissioned blockchain allows defining different levels of exposure for different information sets, e.g., personal information. The IRB data model can be designed to be both private or/and public accessible. As example it is possible to design a data model where the personal information is not exposed or exposed to a subset of peers, while the relationship information can be public exposed and can be easily accessed and used for validation. Besides private data collections [42] provide the opportunity to customize the restricted subset of organizations allowed to access that data. About the usage of a public blockchain to manage resources that are critical for the functioning of the Internet, a potential threat is the loss of the keys that bind these resources. The use of a permissioned blockchain where the organizations are in charge of the certificate issuance, renewal and revocation [37], as every certificate is bind to

a specific resource, provides the same level of control over the managed resource as the one provided by current solutions. Besides the usage of a permissioned consortium model brings a better control over the disputes that may raise in case of an eventual attempt to tamper the system. Indeed the knowledge of the participants identities, together with the high level of traceability offered by blockchains, simplify the eventual dispute resolution. Besides the consortium can decide to disconnect a malicious player.

## 2.2. Background on Internet resource management, routing and RPKI

### 2.2.1. IP Address management

IP addresses identify the end-points of every Internet communication, providing both identity and location functions. An IP address is a 32-bit identifier for IPv4, and a 128-bit identifier for IPv4's intended replacement, IPv6. Each network is assigned one or many ranges of IP addresses (called prefixes), which are non-overlapping with the prefixes assigned to other networks. The administrator of each network then assigns IP addresses to the nodes at the network. The assignment process must consider the limitation of the pools at the time of allocation, and the need to ensure uniqueness and proper registration to meet several operational requirements [43, 44].

In the early days of the Internet, the responsibility for assigning and managing the global pool of IP addresses was performed by the University of Southern California as part of a research project funded by the Defence Advanced Research Projects Agency (DARPA), U.S. government. Between 1998 and 2016, the National Telecommunications and Information Administration (NTIA), part of the Department of Commerce, U.S. government, outsourced the technical management role to the Internet Corporation for Assigned Names and Numbers (ICANN). In 2016, Internet stakeholders, including the U.S. government, agreed to let the contract with the U.S. government expire and ICANN continued to perform the management of the global pool of Internet addresses, numbers and names ever since [45]. Although ICANN's structure seeks for accountability and transparency, ICANN is ultimately subject to the jurisdiction of the California State, in which many lawsuits have been filed [46].

ICANN has delegated the address management duties to the five RIRs, namely AFRINIC (Africa), APNIC (Asia Pacific region), ARIN (mainly US and Canada), LACNIC (Latin America and the Caribbean) and RIPE (Europe, Middle East and Central Asia). RIRs are open membership-based bodies composed primarily of organizations that operate networks. The address resources received from ICANN are assigned according to policies developed

regionally by each RIR, although coordinated with the rest. Then, the resources are allocated to their requesters, according to six goals explicitly agreed among all RIRs in its policies [17, 47–50]:

- **Uniqueness**. Addresses must be globally unique, the "raison d´être" of the registry.

- **Fairness**. Current policies are designed to be fair, in the sense that they should be equally applied to all parties irrespectively of "their location, nationality, size, or any other factor" [17].

- **Conservation**. A main goal of the Internet resource allocation policies is to make a rational use of them and avoid wasteful practices.

- **Aggregation**. The core routers of the Internet networks exchange information about the Internet address space assigned to each network to perform the global routing function. In particular, these routers must store and process advertisements for the prefixes that describe the address space assigned to every network, so the advertisement of a route can be accounted as an *externality* [44]. The lower the number of prefixes exchanged, the lower the hardware requirements imposed to all the routers participating in the interdomain routing system. The allocation policies aim to reduce the number of prefixes advertised by fostering hierarchical allocation to some extent. In particular, allocation policies encourage the use of provider-based address aggregation as a preferred choice, by giving large address blocks of so-called Provider Aggregatable (PA) addresses to network providers, which in turn suballocate them to end users. In this way, the routes to many different end users can be advertised by a single announcement that encompasses the address space of the suballocated blocks, thus reducing the number of entries to store and process in the core routers. RIRs may also perform Provider Independent (PI) assignments directly to end-users, usually smaller, although at the risk that they may be unreachable due to network operators not assuming the cost of routing them [51].

- **Registration**. Contact and other information associated with allocations is stored to help the normal operation of the Internet, such as serving to troubleshoot connectivity incidents. The current Internet Assigned Number Authority (IANA)/RIR based system maintains both a private and a public database with information regarding the allocations: first, any registry allocating an address block keeps (internal) records on the party receiving the resources. This typically involves a contract which includes

detailed contact information. This information is private and it is not used for Internet operations. It can be used for legal purposes as long as the legal actions are valid within the legal context of the host country. In addition, the party obtaining the resources may use any of the available Internet Routing Registry (IRR) to publish contact and technical information related to the resources. However, the information stored in the IRR system is often incomplete or inaccurate [52].

- **Minimized overhead**. The allocation system should work with as little overhead as needed to fulfil its function.

RIRs can allocate PA blocks to Local Internet Registry (LIRs), and they may also provide direct PI assignments directly to end-users. All RIRs define a minimum PA IPv6 allocation of /32 [17, 47–50][3]. The allocations can be (much) larger if the applicant justifies the needs. In particular, the larger allocations so far are /20. The minimum PI allocation is a /48 or a /56 depending on the RIR and they can be larger if justified.

RIRs charge a yearly membership fee to entities holding Internet resources. In all RIRs except for RIPE, fees vary according to the amount of resources received. A /48 PI allocation fee is between US$ 100 and US$ 800, depending on the RIR. A /32 PA allocation fee ranges between US$ 1,000 and US $ 2,500. See figure 2.3 for the detailed information on the fees.

The current fee structure used by the RIRs is not lineal with the number of addresses. The fee for a /32 is roughly one order of magnitude larger than the fee for a /48 while a /32 contains $2^{16}$ more subnets and addresses than a /48. A similar effect can be observed in PA allocations of different size, meaning that the fee for a /20 is significantly less than $2^{12}$ times the fee for a /32.

### 2.2.2. Interdomain routing

The BGP is used to exchange prefix reachability information between the different networks in the Internet. The function the BGP protocol performs is called *interdomain routing*. The different networks participating in the BGP protocol are identified through Autonomous System (AS) numbers, which are 32-bit unique identifiers. The AS numbers are managed in a similar way than IP addresses through ICANN and the RIRs.

The original BGP specification lacks of security features, enabling the unwanted manipulation of routing information. For example, an attacker can advertise someone else's prefix as its own, to hijack the traffic for that prefix. In order to prevent such incidents, the BGP ecosystem has been recently enhanced with origin validation capabilities.

[3] An /32 allocation accounts for $1/2^{32}$ of the address space, and contains up to $2^{32}$ different IPv6 subnets, as each subnet is a /64 prefix in order to accommodate 64-bit identifiers [53]

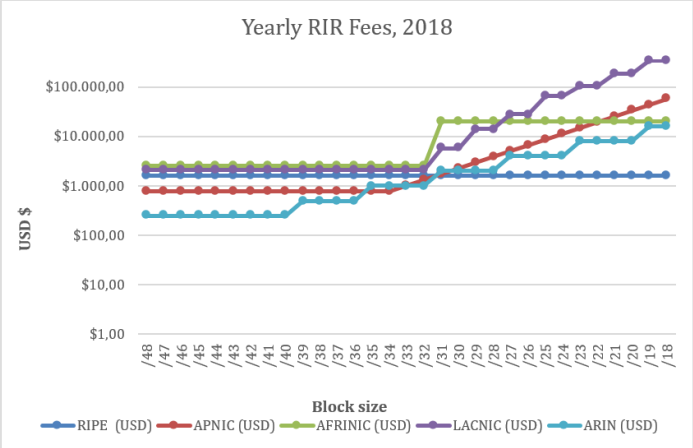


Figure 2.3: Yearly RIR Fees for Provider Aggregatable address assignments, 2018

Origin validation is provided by the Resource Public Key Infrastructure (RPKI) [54] architecture. The RPKI architecture defines a distributed repository that contains Route Origin Authorisation (ROA)s, X.509 certificates that are used to assert that a network, identified by its AS number, is authorized to announce a given prefix BGP (and thus, receive traffic to it). The trust chain of the RPKI starts from the RIRs, that issue certificate delegating prefix ranges to the LIRs , or to the end users, which can in turn issue ROAs. This arrangement is shown in Figure 2.4, in which RIRs issue certificates to LIRs to authorize to sign ROAs for address blocks, and LIRs can further delegate this authorization to end users. The RIRs also provide access to repositories with the cryptographic information issued so far, which at the time of this writing, comprise around 14,000 ROAs [55]. ROAs can be used to discard BGP advertisements, and thus, prevent reachability to certain prefixes, AS numbers, or combinations of both.

Currently, each RIR has its own RPKI root certificate, a self-signed certificate for the whole address space range of IPv4 and IPv6 [56]. A network willing to perform origin validation for the routes to the networks of any region of the world can configure the root certificates of the five RIRs as Trust Anchors, and download the repositories they hold. Then, they use this information to build the router configuration that will filter BGP route information in real time.

[15] shows that certificate issuers can manipulate at will the contents of their publications; for example, they can issue certificates to invalidate or supersede previous resource delegations, affecting connectivity with or through networks performing origin validation. Top-level RPKI entities, i.e., the RIRs, can invalidate or supersede resource allocations they previously allocated, so they can prevent communication for any network with a certificate

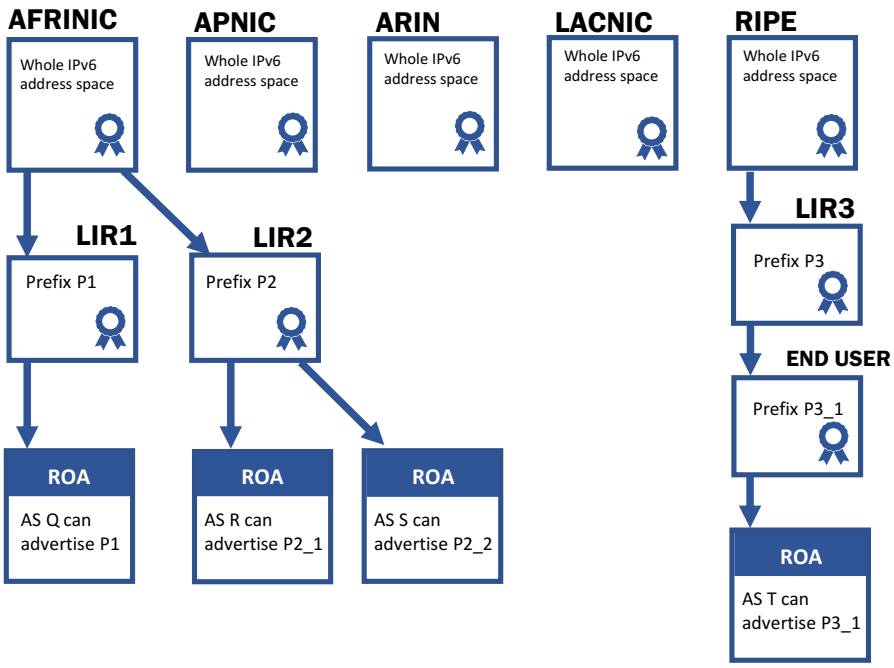


Figure 2.4: Hierarchy of the RPKI and ROAs

rooted in the Trust Anchor of the RIR considered.

Moreover, because each RIR currently has a root certificate for the whole IP address space, any RIR can issue valid certificates for any IP prefix. This means that a RIR can issue a certificate for prefix for which another (valid) certificate, rooted on a different RIR, exists. For example, although prefix P was assigned by RIR_A to an entity in its region, out of its own address pool, and a certificate rooted at RIR_A for prefix P exists, RIR_B may issue a conflicting certificate for the same prefix. The resulting behavior of these actions vary for different third-party networks and routers, and will depend on the configuration of the software applying the validation rules over the RPKIinformation. The network exposed to the conflicting certificates may decide to select one of the certificates (based on explicit preferences on the Trust Anchor, or in the order in which the information has been processed) or discard both. The implications are that the isolated action of a RIR over a prefix can invalidate it for all the networks performing any type of cryptographic validation if the prefix belongs to its region, and can interfere in ways that are hard to predict for prefixes belonging to the address space assigned to other RIRs.

### 2.2.3. BGP and RPKI

BGP is used to exchange prefix reachability information between the different networks in the Internet. The networks participating in the BGP protocol are identified by *AS numbers*, 32-bit unique identifiers. The lack of security features of the original BGP specification has enabled attackers to manipulate the routing information, or propagate legitimately held information in unintended ways. The effects of an attack to BGP include [57]:

- the traffic affected can experience a reduction of the quality of the traffic transmission, even resulting in the complete discard of the traffic (*blackholing*)
- the traffic affected can be manipulated or eavesdropped (*man-in-the-middle*)
- the services of the victim can be impersonated (*imposture*)

An example of one of such an attack occurred in December 2017 [58]. Prefixes usually originated by Google, Apple, Facebook, Microsoft, Twitch, NTT Communications and Riot Games were diverted for three minutes through networks that were not supposed to receive this traffic. Many other attacks against BGP occurred before and after this particular incident. In order to ameliorate the effect of such incidents, the BGP ecosystem has been enhanced with origin and path validation capabilities.

Origin validation is provided by the RPKI [54] architecture. This architecture defines a distributed repository of X.509 certificates used to assert that an entity is the legitimate holder of a set of IP addresses or a set of AS numbers. At the leafs of the RPKI certificate chain we find ROAs, by which the holder of the prefix authorizes a network with an AS number to originate advertisements for it. A ROA is only valid if the whole certificate chain from which it is derived is valid.

The RPKI chain of trust follows the structure defined by the hierarchy of authorities involved in the allocation process for the IP address space and AS numbers. Thus, in order for an End User network to be able to issue a ROA for its prefixes, it needs a certificate issued by the LIR from which it received the addresses; in turn this LIR should have received a certificate from its corresponding RIR.

The chain of trust defined for the RPKI also implies that the RIRs are the natural *Trust Anchors* to configure in order to bootstrap the process of certificate validation.[4]

A router performing origin validation checks if the information included in a BGP route regarding to the AS that originated the announcement of a prefix matches with the content of

[4]Although Trust Anchor selection remains local to the validating network, who may add/remove Trust Anchors at will (for example, to support internal routing policies), in practical terms the anchors of the five RIRs must be configured, or the security information for large regions of the Internet will remain unknown, defeating the purpose of the mechanism.

a valid ROA . The certificate path, and eventually the ROAs, are discarded if the validation fails.

#### 2.2.3.1. Adverse actions against the RPKI

We now describe the set of adverse actions that can be carried out against the different components of the RPKI. In the context of the RPKI, an adverse action is defined as a modification of the information published by the RPKI regarding a set of prefixes that is against the wishes of the holder of the resources. There are different forms of adverse actions [59]:

**Deletion** is the removal of an object from the publication point, against the wishes of the resource holder.

**Suppression** is failing to delete or publish an object according to the resource holder wishes.

**Corruption** is the modification of an object without including a valid signature. The object will not pass the signature validation checks.

**Modification** is the publication of a new valid object that is different to (and replaces) the current version approved by the resource holder.

**Revocation** of a certificate, so that the object will not pass the signature validation checks. This is achieved by including the certificate in the the appropriate Certificate Revocation List, without permission of the resource holder.

**Injection** is the creation of a new valid object into a publication point, without permission of the resource holder.

These actions can be taken against the CA certificates, the ROAs, the manifest and the revocation list. The three first actions can be performed by the publication point of the objects of the resource holders. The latter three actions require compromising the private key of the resource holder. Also, all of them can be executed by a parent CA. For a more detailed description of the possible actions, the reader is referred to [59].

# 3 Related Work

The work related to this thesis can be mainly divided in two macro areas, namely the management of internet resources using blockchain and the BGP route leak protection. In this chapter we present the previous work about the two topics providing a comparison with our solutions.

## 3.1. Management of internet resources using blockchain

We next describe existing work related to the management of Internet resources using blockchain technology.

Namecoin [29] is a working system that provides decentralized namespaces improving security, privacy and censorship resistance. Namecoin supports name resolution for several namespaces, including the Domain Name System (DNS) names within the `.bit` domain. While the `.bit` domains belong to the globally DNS hierarchy, the mechanism for resolving `.bit` names is different than for other DNS names, as it requires querying the Namecoin blockchain for name resolution. This implies changes in the host resolver. Technologically, Namecoin is a fork of Bitcoin with a new native token, the NameCoin Token (NMC). Namecoin then uses a PoW consensus mechanism. The main challenge when using PoW for a new coin is the lack of economic incentives for miners to mine the new coin. In order to address this issue, Namecoin uses merged-mining with Bitcoin. Namecoin DNS names are paid using NMC tokens. The price of the NMC token is defined by the market. In theory, the NMC token price would be high enough to prevent stockpiling. In reality, there was an initial peak on the demand for attractive names such as well-known trademarks. Analysis showed that most of these names belong to three entities, and are not used for name resolution, so it should be assumed that were acquired for speculation. After this initial phase, the price of the NMC token drop to its current value that is close to US$ 0. As a take away, Namecoin is a

valuable experience for designing blockchain based systems for managing Internet resources. First, the Namecoin experience shows that the fee mechanism must use a high-enough fee to be effective. Leaving the market to determine the fee seems unwise, especially in the early phase, when there is little demand, the market laws make the price to decrease, which in turn render the fee mechanism ineffective.

However, there are fundamental differences between IP addresses and DNS names, namely: First, while all IP addresses have the same value, DNS names do not. Some DNS names are more attractive than other (e.g., those reflecting trademarks). Second, in order to use `.bit` names allocated though Namecoin, host modifications are required, while no modification in end hosts are required to use IP addresses allocated through the InBlock.

Several recent proposals (Internet Blockchain [27], IPchain [28], BGPcoin [60], [61]) specifically address the application of blockchain technology to the management of Internet addresses and routing resources. Table 3.1 describes the main characteristics of each of these proposals, along with its comparison with the InBlock implementation.

Regarding to the underlying blockchain used, some works (including InBlock) rely on an existing blockchain (Ethereum), while others propose to create a new one for this purpose.

Internet Blockchain [27] propose the use of blockchains to register IP addresses. They propose the design of a Bitcoin-like blockchain for the management of all Internet resources including IP addresses, ASes, DNS and BGP route. The main goal is to avoid centralization and the control of a single authority. To reduce the design and development risks associated to the blockchain technology, they choose the Bitcoin as its underlying platform. They propose using *multi-sig*, a technology in which a cryptographic operation can be performed or validated with a minimum number of keys from a larger set, to alleviate the key-loss problem. They conclude that this technology is still not mature enough to support path validation, i.e., path authenticaiton for BGP route advertisement. For this reason, they propose an incremental deployment, starting from the management of RPKI functionalities, then supporting route advertisements, and finally providing name resolution (currently solved by the DNS system). Although they consider, as we do, the idea of replacing the source origin validation provided by RPKI with a blockchain-based mechanism, they do not address the design of an automatic address assignment system (with key concerns such as stockpiling prevention or secure bootstrapping).

Palisse et. al. [28] propose a blockchain mechanism to manage IP addresses. They propose to build up a new blockchain to record IP addresses allocation and delegation, based on a PoS consensus mechanism, where addresses are the main asset. The main concern regarding such approach is that PoS results in major stakeholders having more chances to mine new blocks. This means that the current authority of the global IPv6 address pool IANA would

| Mechanism name | Blockchain technology | Resource bootstrap | Initial allocation for LIR | Managed resources | Implementation |
|---|---|---|---|---|---|
| Internet Blockchain [27] | New blockchain, Bitcoin-like | Genesis block includes existing allocations, RIRs register new allocations | Centralized (through RIR) | IPv4 and IPv6 addresses, AS numbers, DNS and BGP routes | NA, only general architecture is defined |
| IPchain [28] | New blockchain, Proof of Stake | RIRs allocate blocks to themselves in genesis block, then register futher updates | Centralized (through RIR) | IP addresses, ROA | Open source |
| BGPcoin [60] | Ethereum | IANA, RIRs, NIRs use their accounts to include their resources | Centralized (through RIR) | IP addresses, AS numbers, ROA, adjacent ASes | Open source (smart contracts) |
| Sfyrakis, Kotronis [61] | New blockchain, Proof of Work | Genesis block includes existing allocations, RIRs register new allocations | Centralized (through RIR) | IP address, AS numbers, ROA, adjacent ASes | Open source |
| InBlock | Ethereum | Genesis block includes IPv6 address block to use in InBlock | Decentralized | Specific subset of IPv6 addresses, ROA | Open source (smart contracts) |

Table 3.1: Comparison among blockchain proposals for management of Internet resources

certainly control the majority of the stakes and probably the blockchain, defeating the goal of preventing centralized control of the IP address allocation system.

Like Namecoin, they propose an expiration mechanism with renewal to prevent the lost-keys problem and to preserve the consistency of the Internet resources. This work does not consider concerns such as stockpiling or secure bootstrapping.

Kuerbis and Mueller [52] discuss the applicability of blockchain technology to perform the IRR functions. Unforgeability and a provable timeline are identified as desirable properties of a routing policy registry, and are naturally provided by blockchains. They consider a system in which operators could cryptographically point to routing policy information repositories by using Blockstack, a blockchain to facilitate decentralized naming systems. Integration with RPKI key data can ensure the authenticity and integrity of the data. InBlock is similar in the sense that it stores the minimal metadata in the blockchain, and points elsewhere to the storage of the information. In this case, the authenticity and integrity of the routing policy data is the cryptographic link to the identity to which the addresses where allocated.

IPchain is based on a PoS consensus mechanism with the addresses being the asset defining the stake balance. The main concern with such approach is that PoS results in major stakeholders having more chances to mine new blocks. While this may work well for IPv4, for IPv6, it means that the current authority of the global IPv6 address pool (IANA) would certainly control the majority of the stakes and probably the blockchain, defeating the goal of preventing centralized control of the IP address allocation system. Sfirakis and Kotronis [61] propose using a new PoW blockchain, but they do not discuss the risks of managing valuable resources as IP addresses in a small mining network.

In all cases, except for InBlock, the resources (IP addresses, ASes, etc.) are initially allocated to LIRs by IANAs/RIRs . We name this approach as centralized, as it depends on the central authorities currently defined for Internet resource management. For the blockchains specifically created for this purpose, existing allocations (e.g., obtained from current registries or the RPKI) can be written in the genesis block. For all other proposals except for InBlock, initial allocations to LIRs have to be performed by IANA/RIRs. For InBlock, IANA/RIRs delegate the IPv6 pool to InBlock, but the assignment to LIRs is decentralized. Because all proposals other than InBlock rely on IANA/RIRs to annotate initial allocations to LIRs, the mechanisms can be used for both IPv4 and IPv6. The decentralized mechanism defined for InBlock requires abundance of resources to allocate, so that it is currently defined for IPv6; it could also be easily extended to ASes, but managing IPv4 addresses would need a different approach.

All the proposals facilitate the transference of the managed resources (or a subset of them) to other LIRs, without the participation of the RIRs or other higher-level entities. They also

enable secure route origin validation by registering the association of the managed prefixes and their corresponding ASes (ROA). In fact, all other proposals but InBlock aim to replace the RPKI for this purpose. As InBlock only manages a subset of the (IPv6) address space, it does not aim to replace the RPKI, but to complement it for cases where decentralisation is desired (besides, InBlock could also coexist with any of this technologies to replace the RPKI for RIR-assigned prefixes).

Additional information such as the list of ASes that are adjacent to a given one can be included in the blockchain to enable limited path validation, such as checking that the AS pairs observed in a BGP route match with the topology annotated in the blockchain ( [60] for the last hop, [27], [61] for the whole the path), although concerns are raised about the ability to perform this in practice [27].

As for the evaluation of the PoC blockchain based implementation we refer to papers [62, 63] that show the design, implementation and measurement of the cost of systems based on Ethereum smart contracts. The main parameter measured in all these system papers is the gas spent in the transactions involved, as it is a key metric to prove its effectively usability in terms of economic cost. In Section 4.4.3 we use the same measurement methodology to provide the amount of gas consumed by each transaction and, after selecting a gas price, their actual cost in dollars. In addition, we include the measurement of mining and confirmation times of the InBlock operations that require writing into the blockchain.

We conclude the section with an overview of the alternative IP Address Allocation Systems that do not employ blockchain technology.

### 3.1.1. Alternative IP Address Allocation Systems

There have been proposals for alternative mechanisms to manage IP address allocations. In particular, there have been several proposals for geographically aggregatable addresses (the original IPv6 address architecture reserved an address range for geographic-based unicast addresses [64]). The main concern with geographic aggregation is that it opposes to the business logic of the Internet Service Providers (ISPs), since it would require ISPs covering a given geographic area to announce the routes for the prefixes of that area and thus carry traffic for customers of the competing ISP in the region. InBlock does not argue for any form of geographical aggregation, and thus, suits to the current business model. Addresses allocated by InBlock can be used as any other Global Unicast Address assigned through the current IANA-RIR based system. Transferable Block Lease (TABL) [65] proposes a market-oriented IPv6 allocation mechanism. They suggest that RIRs set aside an address block that can be allocated without the assessment of the needs to the applicant, with a fee that is related to the size of allocation. The block allocated though TABL would be

transferable between parties. They argue that such an approach would provide a simpler method for address management that would also reduce the provider lock-in problem, as end users would be able to have direct access to addresses that they could keep after a provider switch. Address conservation is enforced by the economic incentives for resource holders to request the amount needed, and trade or return the unused address blocks. They also suggest performing an experiment to assess the benefits and risks of the proposal.

The address assignment mechanism described by TABL is similar to InBlock in the sense that they both rely on fees to access to IP address allocations without an assessment of the address needs of the applicant. They are also related in the analysis of how the resulting mechanism would conserve the address space. However, TABL and InBlock differ in their main objectives: InBlock aims to prevent jurisdictional overflow, while TABL intends to facilitate to ISPs and end users access to address blocks. They also differ in their nature, as TABL is an address assignment policy implemented through contractual framework which still relies in the IANA-RIR for the whole address management process. Consequently, the same entities of the current system (RIRs, National Internet Registry (NIR)) retain full control for registering and issuing the certificates required for RPKI operation. As IANA and the RIRs still control the address management, they can prevent parties to obtain addresses, revoke leases and invalidate its associated RPKI information.

Some of the mechanisms proposed by TABL could be considered for InBlock adoption. Support for address transferability can be implemented as an InBlock function enabling the modification of the account that manages an address allocation. TABL advocates the support for a wider range of address block sizes, any size between /32 and /48. In this case, a large address space block should be reserved for InBlock to ensure for each of the block sizes that a sparse allocation strategy can be used to enable an entity to request a contiguous block to a previously allocated one.

## 3.2. Route leak protection

In this section we are going to focus on other work strictly related to BGP route leak protection.

The current practice to prevent route leaks is the configuration of filters [66]. An ISP should configure inbound filters in the BGP sessions with its peers and customers, so that only routes containing the prefixes and/or ASes allocated to its peers/customers and its peer/customers' customers are accepted while all other routes are filtered out. Frequently, these filters are build using the information available in the IRRs. In order to enable that, the ISP's customers and peers should populate the IRR with the information regarding their

prefixes and ASes and they should also require their own customers to do so. ISPs are sometimes reluctant to take drastic measures such as filtering routes and hence traffic from its (paying) client if the IRR is not populated. With Autonomous System Internet Registry Inference for path Authorization (ASIRIA), ISPs are not required to penalize customers that do not populate the IRR, beyond the lack of leak protection for their prefixes.

[67] extends the BGP OPEN message to explicitly include information about the type of relationship (peer, customer, provider, Route Server, Route Server client) between neighbour ASes. These allows neighboring ASes to verify and enforce that there is mutual agreement on the type of relationship and that the BGP configuration is consistent with the type of relationship. Such extension complements the ASIRIA solution since ASIRIA relies on local information about the type of relationship an AS has with its neighbouring ASes. Having it explicitly announced in the BGP OPEN message would certainly make ASIRIA more robust. [67] also defines a new, optional, transitive, Only to Customer (OTC) attribute that can be used to mark routes that must only be announced downwards to customers. Tagging routes with the attribute allows ASes to detect when the route is leaking and filter it. The approach based on the OTC attribute has several merits, including limited disclosure of routing policy information, which in some cases is considered strategic information. The main advantage that ASIRIA exhibits compared to the OTC approach is regarding initial deployment. ASIRIA relies on existing information in the IRR. So, the first AS to deploy ASIRIA would obtain the protection described in Section 5.5.1, fostering its adoption.

Azimov et all [68] defines a new *path-end record* for the RPKI which includes a "Non Transit Indicator (NTI)" flag. This flag allows ASes to state (in the RPKI) that they are stub ASes and they should only appear at the end of the AS paths in routes announced in BGP. In this way, it prevents the AS setting the flag to generate route leaks. This is an elegant approach that is part of a more general solution to provide Border Gateway Protocol Security (BGPsec). However, the protection of NTI is limited to route leaks generated by stub ASes, while ASIRIA can protect also from leaks generated by other ASes, including any form of customer and peer combinations. Besides, NTI requires populating the RPKI with the new record, while ASIRIA leverages on existing information in the IRR. Also, while it seems apparent that NTI could help preventing route leaks caused by errors, it is unclear how effective they would be against malicious routing leaks, since it is unlikely an AS that intents to maliciously inject a route leak will willingly mark the NTI flag. ASIRIA, on the other hand, can detect leaked routes using the information about the relationships with adjacent ASes, even if the leaker does not collaborate.

There are third party services such as BGPmon[1] and Qrator[2] that detect global route leaks and inform subscribed ASes of such events. The main differences between ASIRIA and those systems are that ASIRIA runs locally, while BGPmon and Qrator are external services. In cases of major disruptions that affect the connectivity, access to third party services may also be compromised. In addition, BGPmon and Qrator use algorithmic inference of relationships, while ASIRIA uses the information declared by ASes in the IRR. This means that each AS has direct control over the information used for inferring relationships and if an AS detects that the relationship is wrongly inferred, it can directly change the IRR and correct it. Such modification is not straightforward in the case of algorithmic inference of relationships.

Additionally we consider previous research work on the application of blockchain technology to route leak prevention. In [69] authors compare IRRs to other methods of governing routing data in a way that enhances internet security, such as RPKI and BGPsec and Routing Security. Besides they consider and remark the opportunity offered by the application of blockchain technology in the data governance problem associated with routing, actually pushing the research in that direction. ISRchain [70] is a blockchain-based inter-domain secure routing framework to manage Internet resources, implemented on Quorum test network using the Raft consensus algorithm. In ISRChain each BGP speaker is required to run a blockchain node. RPL chain [71] is a privacy-preserving route leak protection and trusted execution blockchain based environment implemented on Microsoft Azure. In RLPchain, each AS maintains a global confidential and tamper-proofing inter-domain routing policy repository, using cryptography primitives and detecting and preventing inter-domain routing leaks. Miquel et al. [72] proposes a decentralized approach to prevent routing leaks based on the routing policy repository stored in an HF blockchain. However Miquel's work provides an architectural model in which each AS is required to run an HF peer. All works [70–72] provide a prototype which design architecture requires the run of one blockchain node for each AS. Instead we propose a lighter architectural model where Autonomous System Number (ASN) holders can work as clients of the subset of organization that are in charge of running peers. Indeed we argue that offering the opportunity to join the blockchain as a client to those ASes holder with limited interest in running run a node, would facilitate the adoption of the service resulting in the improvement of routing security. Besides such architectural model based on peers and their clients allows mapping naturally the roles of the current IRR to the IRB.

Finally we discuss Autonomous System Provider Authorization (ASPA) [73] that is a proposal for a new RPKI extension intended to mitigate route leaks. ASPA defines a new

[1] https://www.bgpmon.net/
[2] https://radar.qrator.net/

object for the RPKI that enables an AS to declare its Customer-Provider relationships with neighbouring ASes. ASPA also defines a procedure by which ASPA objects are used to detect leaks by inspecting the AS-path information present in BGP Update messages. The BGP route selection process is modified so that invalid routes are discarded. As many other novel security mechanisms, ASPA faces the challenge of providing value to early adopters. The protection provided by ASPA is directly related to the number of ASPA objects present in the RPKI. During the early phases of adoption there are few ASPA objects in the RPKI, hence the resulting protection will be very limited. This in turn generates little incentives for ASes to deploy ASPA and validate the routes. If we take the deployment of the RPKI as a reference, according to Testart et al. [74], seven years after the completion of the standardisation of the RPKI, only 17% of the routed prefixes were registered in the RPKI, and only 3% of a set of analyzed provider networks enforced the route validation rules. Thus, surmounting the difficulty created by the interdependency between utility and adoption of the proposed solution is critical for gaining widespread adoption of ASPA and hence obtaining Internet-wide protection against route leaks. ASIRIA can help surmounting this challenge and foster the adoption of leak protection mechanisms such as ASPA.

The fundamental difference between ASPA and ASIRIA is the nature of the information used regarding AS relationships. ASPA uses a new RPKI object specifically designed for this, while ASIRIA infers AS relationships from the IRR. While we can expect ASPA information be more reliable and accurate when it becomes available, ASIRIA can rely on existing information, thus providing value to early adopters. We believe that both solutions can be combined, so that ASIRIA can be used while ASPA objects are scarce. Then, some customer provider relationships will be obtained from the RPKI and others from the IRR. In this way, ASIRIA can help bootstrapping ASPA, providing value to early adopters.

In next subsection we provide a detailed overview of the ASPA mechanism.

### 3.2.1. ASPA overview

In ASPA, ASes can declare their (direct) transit providers in the RPKI [75], using the newly defined ASPA record. An ASPA-enabled AS uses this information to verify AS paths of the received routes and detect leaks in realtime.

An ASPA record is a digitally-signed object where the signing AS lists the set of providers authorised to propagate its routes upstream to other providers. It is expressed as ASPA(AS1, AFI, [AS2, AS3, ...]), representing that AS1 declares that AS2, AS3, ... are all its providers for the AFI Address Family.

ASPA defines a route verification process that allows an ASPA-enabled router to determine whether the status of a route is Valid, Invalid or Unknown by analyzing the

AS path attribute of the route. By analyzing pairs of ASes present in the AS path attribute and matching them with ASPA objects present in the RPKI, the ASPA-enabled router can determine if a path is invalid (i.e., if it matches with any of the sequences described as a leak in Section 5.4.1) or it is a valid path otherwise.

ASPA recommends to discard INVALID routes and to accept routes marked as UNKNOWN, enabling ASPA use in scenarios in which the mechanism has not been fully deployed. The outcome of ASPA is accepting or rejecting the route, but it does not affect the preferences of the accepted routes.

In order to analyze the AS path, an ASPA enabled router executes the Customer-Provider Verification (CPV) process to every pair of adjacent AS pairs contained in the AS path.

For an AS (AS1) that may or may not have inserted an ASPA object in the RPKI, ASPA defines the CPV process to validate if another AS (AS2) is a provider of AS1. The possible outcomes of the CPV process for the (AS1, AS2) pair are:

- UNKNOWN, when AS1 has not issued its ASPA record and it is not possible to determine any relationship.
- VALID, meaning that AS2 appears in the ASPA record of A1, so that AS2 is a provider of AS1 and it is authorized to propagate this route to other providers.
- INVALID, to indicate that AS2 does not appear in AS1's ASPA record and AS2 is not a provider of AS1.

The ASPA route verification is different depending on whether the route was announced by a customer of by a provider. If the route is received from a customer or a peer, the path should be an *upstream* one, i.e., it should only include Customer to Provider (C2P) relationships between an AS and the next AS in the path. To check this, the router retrieves the ASPA objects for the ASes in the AS path. It then executes the CPV process to check that each AS declares that the next AS on the path is its provider. If the outcome of the CPV process for any of the pairs is INVALID, the route is identified as a leak and it is marked as INVALID. If no INVALID pairs appear in the AS path, but a mix of VALID and UNKNOWN are obtained, the route is tagged as UNKNOWN. Only if all the pairs are VALID, the route is annotated as VALID. Figure 3.1 illustrates the validation process for an upstream route.

If the route received from a provider, it may have gone upstream for a while, and then downstream. In this case, the route verification process searches for 3 distinct sequences within the AS path, namely, the upstream sequence, the turning point and the downstream sequence. They should appear in that order. In the upstream sequence, one or more consecutive C2P relationships are present. The turning point is either a Peer to Peer (P2P)

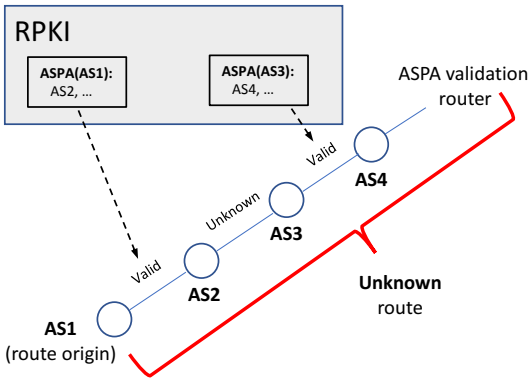


Figure 3.1: Upstream ASPA validation: AS1 and AS3 have registered its providers. As there is no ASPA record for AS2, the route is tagged as UNKNOWN.

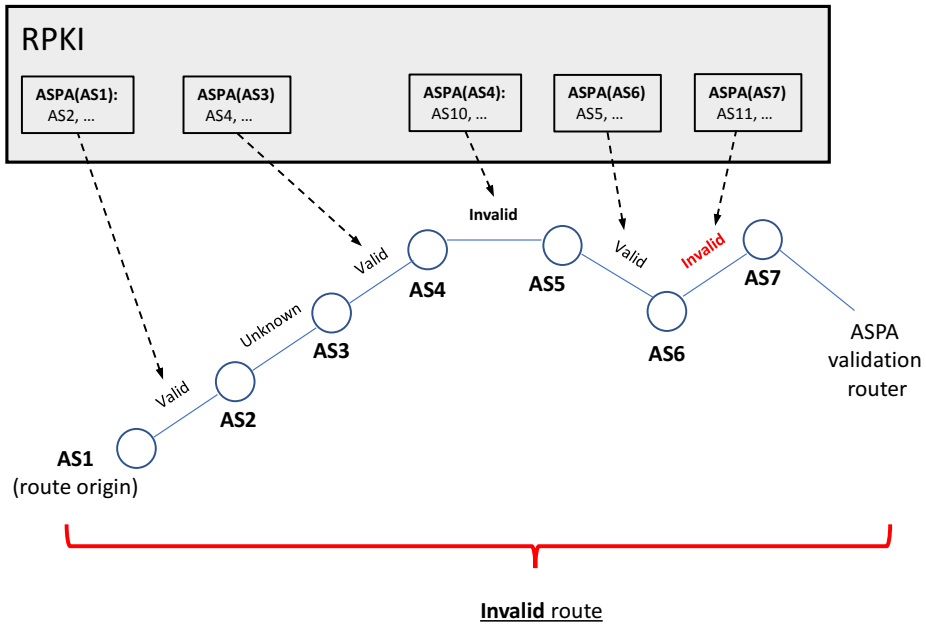


Figure 3.2: Downstream ASPA validation: AS1 and AS3 have registered its providers. The INVALID outcome for the (AS4, AS5) pair indicates that the upstream phase of route propagation has completed. Route check continues with the next AS pairs. In the downstream phase, the AS closer to the destination is checked for a provider indication for the other AS. This check succeeds for (AS5, AS6), but fails for (AS6, AS7), as AS6 is not in the list of AS7 providers. This las INVALID relationship results in the whole route being tagged also as INVALID.

or a Provider to Customer (P2C) relationship. The downstream sequence is composed of one or more C2P relationships. Any other pattern is a leak according to RFC7908.

The ASPA route verification process identifies the three aforementioned sequences using the information available in the ASPA records as follows: The identification of a variable length sequence of C2P relationships from the source of the route is done in the same way as described earlier for a pure upstream case. The first violation of this sequence, i.e., the first (ASx, ASy) pair for which an INVALID value is returned, meaning that ASx has not registered ASy as a provider, is identified as the turning point. The relationship between ASx and ASy may be P2P or P2C. After the turning point, the downstream sequence is identified by searching for a sequence of P2C relationships. This is the same verification as for the upstream case, but made in the opposite direction of the route propagation. That is, for every (ASx, ASy) pair present in the AS path of the route, ASx must be present in the ASPA record of ASy.

If both the upstream and the downstream sequences are composed by VALID pairs, the result is a VALID route. If an INVALID pair is present in addition to the turning point, the route is identified as a leak and marked as INVALID. Else, if there is any UNKNOWN pair in any of the sequences, the route is tagged as UNKNOWN. Figure 3.2 depicts an example in which a downstream route is determined INVALID.

Note that this procedure detects all four types of leaks specified in the previous subsection.

# 4 InBlock

In this chapter we present InBlock. InBlock tackles two main research challenges:

1. Limitations of the hierarchical structure for the Internet resource management
2. RPKI management, potential errors and abuses

In the first section of this chapter we expand the discussion about the challenges faced by this research work. In section two we discuss the general InBlock design, including the different mechanisms that InBlock implements to fulfill the address assignment goals.

Later on in this chapter we present two different proof of concpet of InBlock, namely InBlock6 and InBlock4. In section three we describe the InBlock6 design that provides a distributed, automatic, irrevocable, tamper-free, publicly accessible, privacy-preserving resource allocation mechanism for the IPv6 address space. Lately we present the implementation and the experimental evaluation of the InBlock6 Proof of Concept (PoC). In section four we present the design, the implementation and the experimental evaluation of the InBlock4 PoC. InBlock4 not only inherits InBlock6 functionalities, applied to the IPv4 address space, but it provides an alternative framework to register already assigned resources into the blockchain, enabling the decentralised route origin validation.

## 4.1. Challenges

### 4.1.1. Limitations of the hierarchical structure for the Internet resource management

IP addresses are a cornerstone of the Internet. Every device must configure an IP address in order to be able to send and receive packets through the Internet. As such, the management of the global pool of IP address is of uttermost importance for the correct functioning of the

Internet. IP addresses are globally administered by the Internet Corporation for Assigned Names and Numbers (ICANN), a private non-profit organization. The global pool of IP addresses is managed through a hierarchical structure rooted in ICANN. ICANN delegates ranges of IP addresses to five Regional Internet Registries (RIRs), the second tier of the hierarchy. RIRs allocate address blocks to Local Internet Registry (LIRs), (LIRs, usually Internet Service Providers (ISPs)). End-users normally obtain addresses from the LIRs and in some cases, directly from the RIRs. In some cases, an intermediate agent between the RIR and the LIR exists, a National Internet Registry (NIR), to accommodate the specific Internet resource management needs of a given country.

While this arrangement has generally been successful, it comes with some rough edges. In particular, one recurring concern is that this structure for the management of the global pool of IP addresses results in the jurisdiction of some countries (where ICANN and the RIRs are hosted) overflowing into all the other countries served. ICANN and the RIRs are private organizations that operate within the legal framework of the countries where they are based. However, they each manage the IP addresses for a large set of countries. This implies that for most countries in the world, the management of a critical Internet resource such as IP addresses operates under the legal framework of a foreign country. This also implies that, for most countries, any legal action involving IP address management will be settled in a foreign court of law, making these resources *de facto* subject to laws of a foreign state, as observed in [14]. These concerns have been recently exacerbated by the development of new Internet security techniques, as we describe next.

Over the last few decades, the Internet has become part of the critical infrastructure for most countries. The increasing concern to guarantee its availability has resulted in the design, deployment and adoption of new security tools, such as the Resource Public Key Infrastructure (RPKI) and Border Gateway Protocol Security (BGPsec). These tools aim to provide cryptographic guarantees that whoever is claiming to have an Internet addressing resource is indeed the legitimate holder of the resource according to the defined allocation rules, preventing prefix hijacking attacks and other vulnerabilities. As such, these mechanisms provide the entities up in the hierarchy of the allocation system (i.e., RIRs, NIRs and LIRs) a capability that they lacked so far, namely the capacity to actually enforce the allocations in real time. In particular, they allow entities up in the allocation hierarchy to arbitrarily override an existing IP allocation [15]. So, if/when these cryptographic techniques are widely adopted, the Internet Registries will be able to invalidate allocations, disconnecting whole networks from the Internet, if so dictated by their governing bodies. We note again that a mismatch exists between the geographical scope in which legal, operational and management decisions are taken (the countries where the RIRs are based) and their effects (the whole

world). This situation has raised a number of concerns and it may be one of the reasons behind the lag in the adoption of such technologies. Note that the attacks they are designed to prevent are very real, so security measures to protect the Internet are indeed needed.

The current hierarchical design for managing Internet addresses was probably the most natural one when it was created. IP addresses come from a single global pool and in order to properly perform its function, global uniqueness must be guaranteed, i.e., the system must prevent that the same address is simultaneously allocated to two different parties. When the Internet was designed, the straightforward way of accomplishing this was to rely on a hierarchical structure of organizations to manage the allocations of IP addresses, preventing the allocation of the same resource twice.

However, new opportunities for managing namespaces (in our case, the IP address namespace) surface with the recent introduction of the blockchain technology. Blockchains are distributed databases that are controlled through consensus. By design, blockchains are politically and architecturally decentralized [76], i.e., there is no single entity controlling it, and there is no single point of failure in the infrastructure. As such they prevent any central authority to modify the content of the database. The blockchain technology provides an opportunity to explore alternative IP address management approaches. Moreover, the second generation of blockchain technologies (e.g., Ethereum) allow us to take this approach one step further and explore the possibility to build an autonomous organization that performs the registry functions without any human intervention. The fundamental challenge is how to design the system so that it autonomously performs the registry functions without being subject to abuses and misuses from the (human) users.

### 4.1.2. RPKI management, potential errors and abuses

The Border Gateway Protocol (BGP) is the protocol of choice for exchanging routing information between different administrative domains in the Internet. As such, BGP has become part of the critical infrastructure required for the Internet and by extension of the modern society as whole. For the same reason, BGP has also become a preferred target of attacks directed to derange the normal operation of the Internet. By subverting BGP, attackers can hijack and/or eavesdrop communications as well as execute denial of service attacks. To prevent these attacks the Internet community has developed the Resource Public Key Infrastructure (RPKI [54]) and BGPsec [77]. These tools provide several cryptographic guarantees such as ensuring that the Autonomous System (AS) announcing a route towards an IP prefix is indeed the legitimate holder of the resource according to the allocation rules, preventing prefix hijacking and other vulnerabilities.

The RPKI is a public key infrastructure with a trust hierarchy tree that is aligned with the IP allocation hierarchy. Through the RPKI, the elements in the IP address delegation chain, i.e., Internet Assigned Number Authority (IANA), RIRs, LIRs and end-sites, can obtain certificates that bind prefixes and public keys. The holder of an RPKI certificate can then sign a Route Origin Authorisation (ROA) statement to indicate that a given AS number is entitled to originate a BGP route for that prefix, using the private key associated to a given prefix, and include it in the RPKI. Third parties can validate BGP routes by contrasting the received announcement with the corresponding ROA. In order to prevent routing attacks, routes for which the ROA validation fails are expected to be discarded.

The RPKI standard dates back to 2012 [54]. After eight years during which numerous successful attacks against BGP have been widely reported in the media (e.g., [58, 78]), the deployment of the RPKI is still far from universal. According to the global RPKI deployment monitor[1], more than 80% of prefixes announced in the global routing table cannot be validated using the RPKI. While there are multiple reasons that explain this lag on the RPKI deployment [79], different voices [15, 80] have raised concerns regarding the centralisation effects that the RPKI has on the global routing system. Indeed, under the RPKI model, the entities in the top of the hierarchy may now, through the capabilities provided by the RPKI mechanisms, change at will and in real time the validity of the allocations made by the inferior levels in the hierarchy. As it has been previously shown [81] this opens the door to a number of errors and abuses from higher entities in the hierarchy, with potentially catastrophic consequences for the rest. This centralisation of the enforcement power over the global routing system is a distinctive feature of the RPKI deployment.

Centralisation (a.k.a. consolidation) in the Internet has received a lot of attention lately. There are more and more parties that warn about the impact of centralisation in the Internet architecture, including the Internet Architecture Board (IAB) [82] and the Internet Society [83]. In particular, the IAB has identified four main areas of concerns regarding the centralisation of the Internet infrastructure [84], namely, *reliability* (given that the central entity becomes a *single point of failure*), *surveillance*, *concentration of information* and *extended scope* of the effects of the actions of a few entities. The RPKI brings all these concerns to the routing system. Indeed, the top of the hierarchy now becomes a *single point of failure*, and if compromised can jeopardise the integrity of the internet routing as whole. In the same vein, the top of the RPKI can be abused to modify BGP routes to divert global traffic towards specific eavesdropping points, having both *surveillance* and *concentration of information* implications. The *scope of the effects* of the actions of the top tiers of the RPKI becomes now *global*. Observe that we are not claiming that the entities at the top of the

[1] https://rpki-monitor.antd.nist.gov/

RPKI hierarchy are likely to abuse the power granted to them. We are merely challenging whether from an architectural viewpoint, the centralised approach of the RPKI is the most appropriate one for a global distributed infrastructure such as the Internet.

## 4.2. InBlock design

Hereby, as a solution for the above mentioned problems, we discuss the general design of InBlock, a Distributed Autonomous Organization (DAO) for managing IP addresses. The proposed approach uses blockchain technology to perform IPv6/IPv4 address registry functions. InBlock makes different trade-offs than the current hierarchical allocation system. First and most importantly, InBlock is not centrally controlled, but as any blockchain-based mechanism, it depends on distributed consensus. Second, InBlock operates completely autonomously, without any human intervention. InBlock provides a distributed, automatic, irrevocable, tamper-free, publicly accessible and privacy-preserving resource allocation mechanism, designed with the appropriate (economic) incentives to enforce address conservation. We show that the proposed InBlock design is able to perform the IPv6/IPv4 registry functions in an efficient manner, significantly reducing the operational costs compared to a traditional (human-based) registry and also reducing in orders of magnitude the time required to perform an allocation. In addition, this solution is compatible with the cryptographic security architecture developed for the Internet routing system.

InBlock is a set of programmes that autonomously runs in the blockchain, performing the functions of an IP address registry, as defined by [85]. In a nutshell, InBlock works as follows: InBlock has a block of globally routable IP addresses to allocate. To request an address allocation, an entity transfers a fee to InBlock. Once the fee transaction has been verified, InBlock annotates in the blockchain the prefix allocated to the requester, serving as a ledger of the address assignment for any interested party accessing to the blockchain. Note that, following the paradigm of *code is law*, the organization bylaws are defined in the InBlock code and executed in the same way by any node validating the blockchain. The human action is limited to the initial definition of the allocation rules (before the InBlock is executed in the blockchain). As a blockchain-based organization, it is not controlled by any single entity, so when applied to IP address management, it results in a mechanism that reflects more accurately the current Internet reality as a global network. This approach provides clear and transparent rules without any kind of human discretion, and thus, it reduces legal uncertainty costs for the organizations depending on the Internet for their activities. Address conservation is preserved by a fee mechanism which mimics the current fees charged by the RIRs to the recipients of IP address allocations. In addition, InBlock provides the means

to become the authoritative database in which the assignee of the prefix can associate the basic information for the Internet routing system to operate, currently stored in the Internet Routing Registries [52], along with the cryptographic information that may be used to secure the routing system.

It is worth to note that InBlock is designed as an additional IPv6/IPv4 registry and not as a replacement for the current IANA/RIR based one. The current hierarchical system has stood the test of time, and the ICANN plus RIR system provides services that are appreciated by the Internet community. Therefore, we do not devise a migration strategy in which the information currently hold by the RIRs is transferred to a blockchain based mechanism. We envision InBlock as just an alternative for organizations deeply concerned about the mismatch in jurisdictions, including the RIRs themselves, which may not want to be responsible for address allocations that are subject to legal processes either in their jurisdiction of in the area in which they operate.

As it is the case for most new disrupting technologies, there are many open questions about blockchains including how sustainable are they, how they will evolve, how secure are they, among many others. We now lie on a crossroad: we have a technology that has the potential to bring autonomous decentralized IP address management to the Internet, but it is still too immature to be fully adopted. On one hand, it is in the Internet's genetic code to embrace innovation, but on the other hand, there is too much uncertainty about blockchains and too much at stake to simply give a step forward and adopt an Internet-wide Blockchain-based registry.

For the aforementioned reasons, we believe it would be beneficial for the Internet community to perform a series of experiments on decentralized Internet address management using the blockchain technology. The proposal is to allocate a small IP address block out of the global address space for an experiment and to create one or more blockchain-based organizations to manage the allocations out of that address block, with a predefined lifetime. Such an experiment would enlighten the community with a hands-on experience to inform future directions regarding decentralized Internet address management.

The rest of the design section is devoted to present the rationale for the design of InBlock. For doing this, we analyse each of the requirements for an address allocation system, and how they are fulfilled by InBlock.

### 4.2.1. Description of the proposed InBlock solution

InBlock is a DAO that performs IPv6 address allocation registry functions. By autonomous, we mean that the whole organization lies in the blockchain, in the form of smart contracts running in the blockchain without human intervention. It is decentralized

because the smart contracts run on the nodes that are part of the blockchain. The behavior of the registry is governed by the code of the smart contract. Modifications to the information regarding the registry of IPv6 address blocks are triggered by blockchain transactions and subject to the consensus mechanism of the blockchain. Therefore, the smart contracts define what a valid transaction is and then all the nodes of the blockchain will enforce that only valid transactions modify the address allocation registry information.

InBlock is configured with a block of globally routable IPv6 addresses to allocate. When an entity wants to obtain an address allocation, it uses its Ethereum account to perform a request. The request is basically a blockchain transaction that transfers a predetermined fee (paid in Ether, the Ethereum cryptocurrency) to InBlock. InBlock verifies that the transaction is valid and that the fee has been correctly transferred. Upon reception of the transaction, the InBlock code goes through its state (stored in the blockchain) and finds an address block that is not currently allocated. Once an available block is found, InBlock associates the block with the identity of the entity requesting the block. This allocation information is recorded in the blockchain. The allocations have a predefined lifetime. The holder of the resources can renew the allocation making a new transaction transferring the yearly fee to the InBlock before the expiration date. If this happens, InBlock extends the lifetime of the allocation for another period. Each IPv6 allocation record stored in the blockchain contains the information about the allocated prefix, the holder Ethereum Identity, the expiration date and a pointer where to find additional information about the holder of the allocation, allowing the holder to include contact or other information.

As stated above, InBlock defines the rules that govern the IP address allocation. Phrased in the Internet Registry jargon, this means that the smart contract will encode the IPv6 address allocation policy. It is only natural then that the goals for the design of the InBlock are aligned with the goals of the existent IPv6 address allocation policies. We next describe the different mechanisms that are part of InBlock aimed to fulfill the address assignment goals of uniqueness, registration, aggregation, conservation, fairness and minimum overhead stated by the current RIR-based allocation system (see Section 2.2). As mentioned earlier, the main challenge is to achieve these goals autonomously without mediating any form of human intervention during the process, given that the InBlock's users will be human that will try to misuse and abuse the system. One major concern is how to deter stock piling or other forms of address wasteful practices.

#### 4.2.1.1. Uniqueness

To guarantee uniqueness in the address assignment, we first require that the block of globally routable IPv6 addresses assigned to InBlock is unique (reserved exclusively for this

purpose). Then we rely on the InBlock code, stored state, and the blockchain consensus mechanism to ensure that only unique assignments are valid, and thus, included in the blockchain registry.

#### 4.2.1.2. Conservation

We identify two different concerns affecting conservation namely stockpiling prevention and the reclaim of unused addresses. We describe the mechanisms used by the InBlock to deal with each of them separately.

#### 4.2.1.3. Stockpiling prevention

One major concern to be considered when designing an IP address registry is how to prevent stockpiling, i.e., the accumulation of resources beyond the actual legitimate needs of the requesting entity [17, 47, 48, 50]. IP addresses are valuable assets and given the precedent regarding IPv4 address space exhaustion, some parties may be tempted to obtain IPv6 addresses just in case they become scarce in the future. The InBlock design must provide the means to prevent or at least to control the extent of stockpiling. The current IANA-RIR system *de facto* uses four mechanisms to prevent IPv6 address stockpiling. First, they require a justification for the need of the resources requested, based on planned needs for IP addresses for initial allocations and based on the Host-Density ratio (Host-Density ratio [86]) metrics regarding subsequent allocations. Second, the requirement to become member of the RIR, paying an initial fee, and the charge of a yearly fee to the parties holding resources, in most cases in relation to the amount of resources received (see 2.3). Third, an abundance argument: because there are enough addresses in the IPv6 global pool for all future needs, there is no point for LIRs and end-users to request addresses for the sake of stockpiling. And fourth, the possibility of reclaiming the addresses allocated if the holder of the resources does not comply with the requirements defined in the allocation policy.

In InBlock, we explicitly give up the first and the fourth mechanisms. In order to achieve full de-centralized control, the process of granting a new allocation and the process of renewing an allocation must be fully automatic and encoded in the blockchain. Both the first and the forth aforementioned mechanisms require some form of human intervention, making them incompatible with the InBlock design goals. We argue that the remaining two mechanisms, fees and abundance of addresses, are enough to prevent stockpiling in the IPv6 case.

We next need to define the fee and allocation size structure that suits the InBlock purposes and it is efficient deterring stockpiling. We use as a starting point the current size and fee structure used by the RIRs.

As mentioned in section 2.2, the current fee structure used by the RIRs is not linear with the number of addresses. This poses a challenge for an automated mechanism based in fees to deter address waste such as the one we aim to design for the InBlock. The lower the cost is per address is, the less effective is the mechanism to deter stockpiling and other wasteful practices. On the other hand, if the fee is set to the largest cost per address currently used by the RIRs (e.g., to the cost per address used in /48 Provider Independent (PI) allocations), this would render the cost of a larger impractically high (the cost of a /32 would be tens of millions of US$ is the cost per address of a /48 is used).

It is challenging for the InBlock to have different cost per address depending on the size of the allocation, because this may encourage applications for larger blocks even when not needed, resulting in address waste (note that we do not have a complementary mechanism such as a need assessment, to modulate user requests). On the same vein, having larger allocations with a low cost per address may promote a secondary market, where it is possible to obtain larger allocation for the same fee (or less) that it would take to obtain a smaller allocation directly from the InBlock[2]. This would again create the incentives for applicants to obtain larger allocation than what they would really need through the secondary market, resulting yet again in address waste. In figure 4.1 we depict the distribution of block sizes of

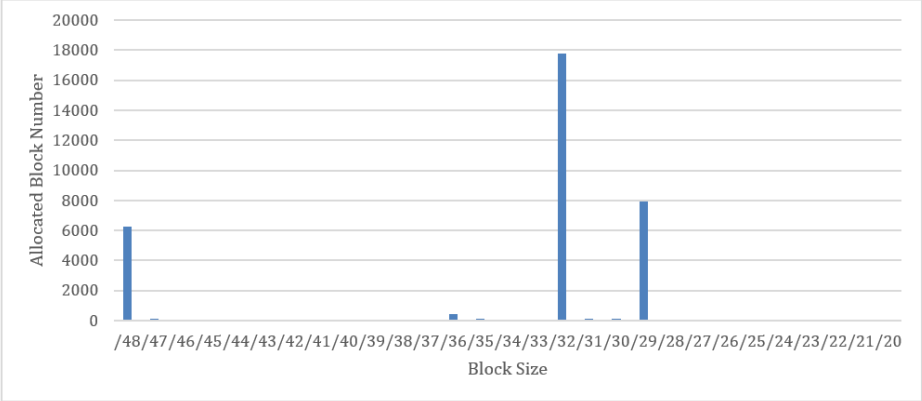


Figure 4.1: Distribution of number of blocks allocated up to May 2018, per block size.

all the existent allocations (for all the RIRs[3]). We observe that, at the time of this writing, there are 17,795 allocations of /32, 6,283 allocations of /48, 7,903 allocations of /29. There

[2]Note that this is a risk of the current fee system, in which the cost grows in a sublinear trend with the number of addresses.

[3]ftp://ftp.afrinic.net/pub/stats/afrinic/delegated-afrinic-extended-latest
ftp://ftp.apnic.net/pub/stats/apnic/delegated-apnic-extended-latest
ftp://ftp.arin.net/pub/stats/arin/delegated-arin-extended-latest
ftp://ftp.lacnic.net/pub/stats/lacnic/delegated-lacnic-extended-latest
ftp://ftp.ripe.net/pub/stats/ripencc/delegated-ripencc-extended-latest

are 191 allocations larger than /29. So, roughly half of the existent allocations are /32s, 25% are /29s and the remaining 25% are PI allocations of /48.

Considering all the above, we propose that the InBlock only allocates /32s and /48s, charging a fee similar to the one found in the RIRs. By doing this, we can satisfy the most common allocation sizes. If an entity requires more than a /32, it probably can afford to request 2 or even 4 /32s, paying the corresponding fee for each of them. Since the fee increases linearly with the number of addresses for allocations larger than a /32, we believe that this fee structure would be enough to deter request additional /32 blocks that are not really needed.

This still does not address the problem regarding wasteful allocations involving address needs smaller than /32. Current RIRs fees are such that a /32 costs in the range of US$ 2,500 to US$ 1,000, and a /48 PI costs in the range of US$ 100 and US$ 800 (plus an additional initial fee in the range of US$ 250 to US$ 2,500). Suppose that the InBlock charges US$ 2,500 for a /32 and US$ 300 for a /48. Since the fee is one order of magnitude larger for a /32 than for a /48, this is likely to be enough to avoid the majority of applicants that would satisfy their needs with a /48 to request a /32, reducing address waste.

A final argument to support that the proposed fee structure is sufficient to deter stockpiling is the following: If the fee of a /32 is 2,500 US$, then getting the whole IPv6 address space would imply a total amount of $10.5 * 10^{12}$ US$ (compared to the world Gross Domestic Product, Gross Domestic Product (GDP) which is $80 * 10^{12}$ US$ for 2017 [87]), making it impossible for any party to even get hold of a significant chunk of the IPv6 address space.

Some further considerations about the fees:

- **Currency**. InBlock will define the fee in a fiat currency (Euros, US$, Yuans) but the InBlock will actually collect the fee in the cryptocurrency used in the blockchain (Ether in Ethereum). Because the prices of the cryptocurrencies fluctuate significantly, we propose to define the fee in the fiat currency and convert the fee from the fiat currency to the current correspondence in Ether.

- **Fee Update**. Fees cannot be constant because if the fiat currency value devaluates, the fee will be less effective as a mechanism to deter stockpiling. On the other hand, InBlock is designed to work without human interaction. We propose to link the evolution of the fee to the increase of the world GDP. In this way, the fee will be updated to preserve its value in the future. Also, making the update of the fee automatic (and preventing any human interaction when defining the fees in the future) provides a stable framework for parties holding resources and prevents from having a human deciding on the fees (which may give the human the power to arbitrarily increase

the fee, and to push the parties out of the system).

- **Destination of the fee money**. It is worth to note that the fee, in the ranges discussed above, is just a mechanism to prevent stockpiling. The actual cost of running the InBlock is way much less than the fee (we estimate 15 US$, see 2.1.2). Several institutions involved in the resources (ICANN, RIRs, Internet Engineering Task Force (IETF), Internet Society (ISOC)) could be natural recipients of this money. We note that the actual destination of the money is not relevant for the mechanism, as it would even work if the money were just destroyed (sent to an non-existent account).

#### 4.2.1.4. Reclaiming unused addresses

With the exhaustion of IPv4 addresses, attention has been paid to policies to reclaim unused addresses. However, only market-based approaches have proven to be effective, as coercive measures face many challenges [44, 88]. The InBlock expiration model provides a mechanism to reclaim unused addresses. When an address allocation is not renewed before its expiration date, the address block returns to the pool and the address space is eligible for further reassignment.

Besides, this mechanism serves to fix a well-known issue with storing tokens in blockchains: if the holder of the resource loses the private key that secures the allocation, the resource is lost. In particular, it is estimated that 20% of existing Bitcoins are "lost" because of this [89].

#### 4.2.1.5. Aggregation

Another important consideration that an allocation mechanism needs to address is related to the preservation of the global routing table. Due to the large size of the IPv6 address space, fostering aggregation is crucial for the viability of the routing system. Current RIR allocation policies promote aggregation through the preferred use of Provider Aggregatable (PA) addresses, but they still allow PI allocations. PI allocations, usually much smaller than PA, contribute to the size of the global routing tables, as they cannot be aggregated. In the previous section we presented a fee structure to deter parties from requesting larger allocations than needed. In this section, we discuss the factors that hinder the deaggregation resulting from a secondary market of address blocks. We also present how InBlock enables aggregation of multiple address blocks obtained by the same entity through a sparse allocation policy.

#### 4.2.1.6. Deagggregation of address blocks due to secondary market transfers

One potential concern with the proposed fee structure is that it may stimulate the creation of a secondary address market that may extend the use of PI allocations by end sites as opposed to PA allocation from the LIRs.

Currently RIRs charge a fee for each PI allocation. While the current policies do impose a number of requirements on end sites requesting PI block, these requirements are usually in the form of a plan or a prevision regarding number of hosts/sites in the near future. We argue that because the InBlock charges a similar fee than the RIR for a PI /48 allocation, it is unlikely that there will be an increased demand of PI allocations due to the InBlock.

The current fee structure would make economically attractive for a party to obtain a /32 from the InBlock and re-sell smaller allocations cheaper than the corresponding fee from the InBlock. This may render PI-sized allocations more affordable, and parties that would not be willing to pay the yearly fee that the RIR or the InBlock charges for a /48 allocations, may be willing to obtain a much cheaper /48 from this secondary market. Since these would indeed be PI allocations (as they are not provided by the ISP), they are likely to be announced in the global routing table as separate routes, bloating the routing table. Note that PI allocations are attractive because they avoid provider lock-in and the associated renumbering cost if the customer changes ISP.

However, a /48 obtained in the secondary market is not a perfect substitute of a /48 obtained directly from the InBlock. The reason is that there is an intermediary in the /48 obtained through the secondary market, the reseller. If the intermediary fails to renew the /32 where the /48 are extracted from (for example, because it goes bankrupt), all the /48 allocations will not be renewed and the end sites will lose their addresses. Moreover, once the end site has obtained an allocation from the intermediary and configured in their network, the intermediary has an incentive to increase the charged fee since there is a cost from the end user to renumber its network. The end site would then be trading ISP lock-in for intermediary lock-in.

Nevertheless, it is still possible that some sites find it attractive to obtain PI blocks in the secondary market, and negatively impacting the global routing table. In order to prevent this, the InBlock could limit the number of different AS numbers used in ROAs within a single /32 to a maximum number (e.g., 100 different AS numbers). This restriction would not impose a real restriction in the operations of ISPs using the /32 for PA allocations, but would negatively affect the intermediaries that want to resell PI allocations out of a /32, since it would prevent all the end sites obtaining an allocation out of a single /32 to use different origin ASes in their ROAs (which is what they would naturally do when announcing a PI

block in the interdomain routing).

#### 4.2.1.7. Sparse allocation

In order for the multiple blocks assigned to a single entity to be aggregatable, the InBlock will use a sparse allocation strategy [90] to manage the overall pools. This allows the holder of a resource to request for the contiguous block, so that the multiple blocks that a given entity obtains from the InBlock are aggregatable.

When submitting a new request for a new block, the applicant can attach a proof that it holds a block from a previous allocation. If this is the case, InBlock will allocate a contiguous block, enabling the aggregation of the two prefixes.

#### 4.2.1.8. Registration

In the case of the InBlock, blockchain identities are mostly anonymous[4]. Moreover, payments are done using the corresponding Ether, which as of today provides significant anonymity features. All this implies that the InBlock has no information about the entity that received the allocation. InBlock provides the means to voluntarily include contact information or route policies for each allocation, in a similar way to Internet Routing Registries (IRRs) [52], but it does not mandate it. However, it is worth to note that the blockchain enables a cryptographic link between the holder of the account to which the address prefix has been assigned, and the routing information. This means that any other party accessing to the blockchain can verify that the contact and routing policy information is authorized by the assignee of the Internet resources.

In summary, InBlock provides stronger privacy features than the current IANA-RIR system, while still enabling resource holders to voluntarily provide contact and routing information for operational purposes.

#### 4.2.1.9. Fairness

Fairness is an explicit goal in current RIR address allocations policies. Fairness in this context means that the policies should be equally applied to all parties irrespectively of "their location, nationality, size, or any other factor" [17]. InBlock naturally achieves that goal, since any party can obtain an Ethereum identity and then it can obtain an allocation from the InBlock. Moreover, InBlock takes a step further, as it achieves jurisdictional fairness. As

[4]Blockchain identities are not perfectly anonymous, since there are means to try to link an Ethereum identity to a physical entity, but in general, it is not required prior identification to obtain a Ethereum identity.

stated earlier, one explicit goal of the InBlock design is to prevent the so-called jurisdictional overflow, where the allocations of entities in one country are under the jurisdiction of another country. We can phrase this in terms of fairness, i.e., that every entity has the right that its address allocations are not ruled by the legal framework of another country. This is not the case today, since allocations to entities based in the countries where the RIRs are based are ruled by their national law system, while for allocation to all other entities are ruled by foreign legal systems.

In particular, we identified the capability of entities placed in higher levels of the RPKI hierarchy of revoking existing allocations as a major concern springing from the jurisdictional overflow problem (e.g., a court of law of the country where a RIR is based can revoke an allocation obtained by an entity based in another country served by the RIR). With the InBlock approach, this problem no longer exists because, given the immutability of the underlying blockchain technology, there is no single entity capable of revoking an allocation. In other words, blockchain technology enables the creation of self-imposed restrictions in the management of a database. In the case of InBlock, one of such restrictions is that no entity can revoke an existent allocation. By giving up the revocation capability altogether, we provide certainty to prefix holders that no entity will be able to interfere with their address allocations.

#### 4.2.1.10. Minimized overhead

By all accounts, the InBlock operation is very efficient and provides reduced overhead. As there are no humans involved in the operation, the costs are very low even considering the maintenance costs of the blockchain itself (about US$ 15, see section 2.1.2). Also, the time-scale of operation of the InBlock is significantly shorter than the current system. Allocations in InBlock are completed in the order of minutes.

### 4.2.2. Limitations and open issues

Now we discuss some limitations of the proposed InBlock design that result from its automatic nature.

InBlock lacks the strong traceability features available in the IANA-RIR system for lawful purposes. Contact and other information associated to the allocations registered by the InBlock are only performed on a purely voluntarily basis. This implies that misbehaving entities are unlikely to include any information that would be useful for their traceability. Networks participating in different types of attacks (denial of service, impersonation, etc.), should be identified through other means. We argue that this is an appropriate approach

and that the traceability for legal reasons should be pursued through the ISPs providing connectivity to the allocated address block. The reason why we find this appropriate is that the InBlock provides a global service hence it should avoid the asymmetry existing in the current IANA-RIR system between the country hosting the registry and the rest of the countries. Pursuing legal actions through the ISPs providing connectivity to the allocated prefix seems to result in a better match between the scope of the organization and the scope of the legal framework.

One of the well-known shortcomings of cryptocurrencies is the dependency on the access to the private key associated to the account holding the tokens. In the case of InBlock, we have addressed this problem by requiring yearly renewal of the address allocation. In case the private key is lost, the address block would not be renewed, and the addresses will be returned to the pool. However, this will impose for the network previously using these addresses to acquire a new address block, and renumber its network, which is known to be expensive [91]. Moreover, if the private key is stolen (instead of lost) the robber can start using the addresses as his own (e.g., generating a new ROA), and there is no higher entity that the victim can appeal to. Note that this is a fundamental property of a system in which the ownership proof is tied to the control of a private key.

InBlock design brings predictability to address requesters, but it restricts the capability to react to unforeseen situations. For example, InBlock ties the evolution of the address allocation fee to the GDP. If due to some unexpected event there is a need to increase the fees this would be possible for new instances of the InBlock but not for existing ones. On the other hand, if the cost of the fees decrease, networks with addresses obtained through InBlock would face the dilemma of acquiring a new address block and experience the renumbering costs, or keep paying the higher fee derived by InBlock for the same resource.

### 4.2.3. InBlock security analysis

In this section, we analyse different attacks that can be performed against InBlock. InBlock does not propose a new security method. Instead, InBlock relies on existent security methods, i.e., the Ethereum blockchain, with well-known security properties, to provide functions currently supplied by the RPKI. So, in order to analyze InBlock from a security perspective, we follow the attack analysis framework presented in Section 2.2.3.1 for the RPKI and compare it to the security properties of Ethereum established in the literature. As described earlier, there are six types of adverse actions that can be executed against a secure IP address attestation, namely, *Deletion*, *Suppression*, *Corruption*, *Modification*, *Revocation* and *Injection*.

In table 4.1 we compare the means required to execute the different types of adverse

actions against the RPKI and the InBlock. The fundamental difference is that in the case of the RPKI all these actions can be performed by a parent Certificate Authority (CA), while this is not possible in the InBlock, as InBlock does not rely on a hierarchical structure. Avoiding the potential errors, abuses and misuses incurred by a parent CA is the main design goal of InBlock. In addition, regarding the first three types of actions, the difference is that blockchain information publication is not limited to a reduced set of publication points, but all blockchain nodes propagate the blockchain information. We next present in detail the different attacks against the InBlock.

**Deletion**: In the context of InBlock, we define the *Deletion adverse action* as the removal of information contained in a confirmed block.[5] Removal of information that was included in an unconfirmed block is considered a *Suppression adverse action* and discussed in the next paragraph. In InBlock, because of the very nature of the underlying blockchain technology, the Deletion attack is essentially unfeasible without affecting the whole blockchain. Once the data is stored in a confirmed block in the blockchain, it is immutable.[6] In order for the data to be actually removed, all Ethereum full nodes (from which the blockchain can be retrieved) must be compromised. At the time of this writing, there are 7,530 active full nodes in the Ethereum blockchain.[7] An InBlock third party can connect to any of these to obtain a copy of the InBlock data. In the case of the RPKI, there are 14 repositories to retrieve the RPKI data from.[8] While each RPKI item is associated to a single authoritative publication point, other repositories can store copies [54].

| **Adverse action** | **RPKI** | **InBlock LIR** |
|---|---:|---:|
| Deletion | Pub. Point, Parent CA | Ethereum nodes |
| Suppression | Pub. Point, Parent CA | 51% attack |
| Corruption | Pub. Point, Parent CA | Ethereum nodes |
| Modification | Priv. Key, Parent CA | Priv. Key |
| Revocation | Priv. Key, Parent CA | Priv. Key |
| Injection | Priv. Key, Parent CA | Priv. Key |

Table 4.1: Comparison of adverse actions against the RPKI and the InBlock

[5] In Ethereum, a block is considered confirmed after 12 additional blocks have been added to the blockchain

[6] The modification the data stored in the blockchain requires a *fork* of the blockchain. Forking is a public event that forces all Ethereum nodes to actively position themselves regarding the adoption of the fork and third parties would act accordingly.

[7] https://www.ethernodes.org/sync

[8] To obtain the number of different RPKI repositories, we configured an FRRouting (FRR) router, https://frrouting.org, to gather the existing RPKI information and we parsed the logs resulting from this operation. The RPKI validator connects to the repositories associated to the five trust anchors (AFRINIC, APNIC, ARIN, LACNIC, RIPE), and retrieves the information stored and pointers to other repositories. The total number of repositories observed by January 2020 is 14.

**Suppression**: This action can be achieved if the attacker controls at least 51% of the mining power of Ethereum [18]. If this the case, the attacker can prevent new transactions to be included in the blockchain by using its hashing power to mine new blocks on top of blocks that do not contain the victim's transactions and discard those that do. Currently, the hash rate of the Ethereum network is 176TH/s.[9] The cost of acquiring the computational power to perform a 51% attack against the Ethereum network is estimated as US$ 136,982 per hour[10], and the effort should be kept for the time during which the information is to be suppressed.

**Corruption**: Because every Ethereum full node validates all the transactions included in all the blocks before propagating the new block, performing this action would require to compromise all full nodes in the blockchain to publish invalid information, so the analysis is similar to the one for the Deletion attack presented earlier. Note that the corrupted transaction is stored in the blockchain, and this will become apparent to any third party auditing it.

**Modification**: To add a new transaction that modifies the information of an attestation contained in the InBlock, the attacker needs to compromise the private key of the affected LIR. The effort to compromise the private key of the holder of the resources is equal in the InBlock and the RPKI, the difference being that in the case of the RPKI, it is also subject to the vulnerabilities involving the parent CA.

**Revocation**: To revoke an existing attestation in the InBlock, the attacker must compromise the LIR's private key. The vulnerabilities of the parent CA also affect the RPKI in this case.

**Injection**: Similarly to the two previous attacks, this is feasible only by compromising the LIR's private key. The vulnerabilities of the parent CA also affect the RPKI in this case.

[9] https://www.crypto51.app/
[10] https://www.crypto51.app/

## 4.3. InBlock6

InBlock supports both IPv4 and IPv6 address space. For this Proof of Concept (PoC) implementation we focus on IPv6 as IPv6 has a large remaining pool of unassigned addresses, while the vast majority of the IPv4 address space has already been assigned [88]. InBlock6 is implemented as a set of Smart Contracts in Ethereum. InBlock6 runs autonomously, i.e., without humans involved in its daily operation.

Next we describe the implementation of the InBlock6 PoC and we show how InBlock6 behaves identifying the roles of the actors involved and we detail how basic operations proceed. Finally we evaluate its performances through an experimental analysis.

### 4.3.1. Overview of InBlock6's operation

InBlock6 is configured with a block of globally routable IPv6 addresses to allocate. This is done at the moment of the deployment of the InBlock6 smart contract and cannot be modified afterwards. Once deployed, InBlock6s6 runs autonomously, processing address allocation requests received and managing the available address pool. Additional InBlock6 instances can be deployed with different address blocks to allocate, so that each instance independently manages a mutually disjoint block of addresses.

When an entity (called an *InBlock6 Local Internet Registry (LIR)* wants to obtain an address allocation from the InBlock6, it uses its Ethereum account to perform a request. The request is in the form of a blockchain transaction that transfers an allocation fee (paid in Ether) to InBlock6. InBlock6 verifies that the transaction is valid and that the fee has been correctly transferred.

All the prefixes allocated by an InBlock6 instance have the same size. However, InBlock6 supports *aggregatable allocations*, i.e., a requesting party may ask for multiple prefix allocations that can be aggregated in a larger prefix.

The allocation fee is significantly larger than the actual costs of the InBlock6 operation. As we present later on, the operational cost (i.e., the miner's fees to perform the transactions required to annotate the allocation in the blockchain) is in the order of a few US$, while the allocation fee is expected to be similar to the cost of an allocation in the Regional Internet Registry (RIR) system, around several hundreds of US$. The reason for this is that the fee serves as a mechanism to deter address stockpiling and other wasteful practices. The reader is referred to [1] for a thorough discussion on prefix allocation sizes and fees.

Upon reception of the transaction, the InBlock6 code goes through its associated state (stored in the blockchain) and finds an address prefix that is not currently allocated. Once an available prefix is found, InBlock6 associates the prefix with the identity of the requester,

its Ethereum identity. This allocation information is recorded in the blockchain.

Allocations have a predefined lifetime. The holder of the resources can renew the allocation by making a new transaction in which it transfers the yearly fee to the InBlock6 before the expiration date. If this happens, InBlock6 extends the lifetime of the allocation for another one-year period. Each IPv6 allocation record stored in the blockchain contains the information about the allocated prefix, the holder's Ethereum Identity, the expiration date and a pointer to additional information, such as the holder's contact or routing policy information.

Once an InBlock6 LIR has obtained an address allocation from the InBlock6, it can suballocate part of this prefix to other entities (called *InBlock6 End Users*). In this way, the InBlock6 mimics the RIR allocation chain, from InBlock6 to InBlock6 LIRs and to InBlock6 End Users. All entities that have obtained an InBlock6 prefix can also include Route Origin Authorisation (ROA) information for it, enabling route origin validation by third parties based on the information stored in the InBlock6.

#### 4.3.1.1. InBlock6 roles

We next describe the different roles defined for InBlock6 and their associated operations.

**InBlock6 Manager**. The InBlock6 manager inserts the InBlock6 code into the Ethereum blockchain by means of a transaction. Then, it issues another transaction to activate the InBlock6 instance, in which it defines the address pool available for the InBlock6 to allocate, the size of the allocations and the allocation fee in dollars. The InBlock6 manager also manages the Ethereum account that is used to receive the allocation fees, so that it can transfer the funds received through the InBlock6 account to any destination account.

InBlock6 does not provide functions to allow the manager to create or modify prefix allocations once the contract has been activated. The InBlock6 manager cannot revoke existent allocations nor it can prevent any party to issue a transaction to obtain a new allocation or to renew an existent one. The InBlock6 manager cannot modify the allocation fees neither. The motivation for this is to prevent the manager from arbitrarily raising the prefix allocation fees in order to impede particular entities to obtain an address allocation. The allocation fee is automatically updated by the InBlock6 contract according to the world's gross domestic product, to account for global inflation. All these restrictions serve the purpose of limiting the power of the InBlock6 manager over the system, drastically reducing the centralisation of power in the allocation system.

**InBlock6 LIR**. An InBlock6 LIR is an entity that obtains a prefix allocation directly from the InBlock6. To do so, the InBlock6 LIR issues a transaction that transfers the allocation fee to the InBlock6. While the prefix allocation fee is defined in a fiat currency (e.g., US$),

the Ethereum transactions transfer Ether. Historically, the exchange rate between these two has varied greatly in time. So, the InBlock6 LIR first determines the amount of Ether that corresponds to the prefix allocation fee. This is done through the use of oracles.[11] The InBlock6 LIR issues a first transaction by which the InBlock6 code writes in the blockchain the currency exchange rate information retrieved from the oracles. Then a second transaction is issued by the LIR through which the InBlock6 code computes the allocation fee (expressed in the fiat currency) and stores it in the blockchain. This fee value is valid only for 24 hours for the requester account. Once the value in Ether of the allocation fee has been determined, the LIR issues the transaction that transfers the said amount of Ether to the InBlock6, requesting the allocation of a prefix.

Upon the reception of the transaction, the InBlock6 code selects a free IPv6 prefix from the InBlock6 address pool, and associates the prefix to the account of the LIR. If the LIR already has a prefix allocated, it can request that the additional address space allocated is contiguous to previously owned allocations, if such address space is available. The prefix allocation is valid for a year and the LIR has to pay a yearly fee to renew it. If the prefix is not renewed, it returns to the InBlock6 free address pool for future allocations.

The LIR is entitled to manage its prefix allocations, so that it can include ROAs, contact and routing information associated to its prefixes (or subprefixes) in the blockchain. The InBlock6 provides functions to enable the LIR to assign sub-prefixes to other accounts (InBlock6 End Users). These users can autonomously update the contact and routing information for these delegated prefixes. The InBlock6 LIR also has the capability to revoke a prefix assignment to an InBlock6 End User.

**InBlock6 End User**. We refer to the entities which have received a prefix assignment from InBlock6 LIRs as *InBlock6 End Users*. We expect that a common case will be that InBlock6 LIRs are Internet Service Providers (ISPs) and that the InBlock6 End Users will be the ISP's customers. In this case, the address assignment from the InBlock6 LIR to the InBlock6 End User will be part of the provisioning of the Internet access service from the ISP to the customer. InBlock6 does provide the means to annotate information about the InBlock6 End Users in the blockchain though. InBlock6 End Users are authorized to update the InBlock6 registries associated with their assigned prefixes with contact and routing information. In particular, InBlock6 End Users can include ROAs that indicate the numbers of the Autonomous System Numbers (ASes) authorized to originate a Border Gateway Protocol (BGP) route advertisement for the prefix.

**InBlock6 Third Parties**. InBlock6 Third Parties access to the InBlock6 information stored in the blockchain to validate routes. A Third Party accesses the Ethereum blockchain

[11] Oracles are third party services that write in the blockchain data from the external world.

and configures the InBlock6 identifier contract as a Trust Anchor for the validation of the information related with the prefixes managed by InBlock6, meaning that the Third Party trusts in InBlock6 for the management of these prefixes.

InBlock6 Third Party validation is performed through Ethereum calls. The calls encode and enforce the validation rules defined in the InBlock6. For example, the InBlock6 defines a call to verify if a given AS number is allowed to originate a route for a given prefix (i.e., if there is a valid ROA for a *'prefix, AS number'* couple). This call executes the following verifications. First, it verifies the allocation chain to ensure that the prefix belongs to the InBlock6 address pool and that it has been allocated to a LIR and that the LIR has assigned it to an End User. Once the allocation chain is verified, it obtains the ROA information from the blockchain and verifies that it contains the requested AS number.

Although these checks could be left for implementation to interested parties, we believe that the provision in the InBlock6 itself of the code that performs all the intended checks provides an additional level of security and compliance. The validation of the AS information according to the assignment, lifespan and delegation rules encoded in the InBlock6 extends the 'code-is-law' motto from the state creation to its validation. Besides, users of these calls do not need to be aware of the inner InBlock6 machinery to make use of its information.

### 4.3.2. Implementation

We have implemented InBlock as a set of smart contracts (a main smart contract and several utilities implemented in different contracts for convenience) written in Solidity, and deployed it in Ethereum. We next describe different relevant aspects of our implementation.

#### 4.3.2.1. Activating InBlock6

To activate an InBlock6 Smart Contract previously inserted in the Ethereum blockchain, the InBlock6 Manager issues an `activateInBlock6` transaction. This transaction defines the InBlock6 pool of prefixes, in the form of a (large) prefix, the length of the prefixes to allocate, and the prefix allocation fee. The fee is set in a fiat currency and its value is updated yearly by the InBlock6 according to the variation of the world gross domestic product, Gross Domestic Product (GDP), measured over the last 10-year period, i.e., 3.3% for 2008/2018.

#### 4.3.2.2. Allocating prefixes

Any entity can request a prefix from InBlock6, thus becoming an InBlock6 LIR. The prefix allocation process is performed in two steps: a first phase which comprises two transactions and one call to determine the value in Ether of the prefix allocation fee, and a second phase

to actually allocate the prefix to the LIR, accomplished in one transaction. InBlock6 relies on several third parties, *oracles*, for obtaining the conversion rate between the fiat currency and Ether. An oracle is a service provided by a third party that annotates in the blockchain a particular external world value. Once ensured that the information was inserted by the trusted oracle, this value can be used by a smart contract. *Provable* [92] is a solution to provide a secure connection between Ethereum smart contracts and the external world. In order to avoid dependency with a single oracle, InBlock6 uses several (three) different oracles providing currency conversion. The risk of individual oracles being compromised is reduced by removing outlier values: if the three oracles are available, the median value is returned; with two oracles, the mean is used. If the all the oracles go offline, InBlock6 uses a default value set in deployment time.

The `getOracleCurrencyConversion` transaction is issued by the LIR and triggers the InBlock6's query to the oracles to obtain the exchange rate. Once the exchange rate is retrieved, the LIR issues the `computeAllocationCost` transaction to determine the allocation fee in Ether. The calculated fee is valid for 24 hours.

With the fee value, the LIR can select one of two different types of transactions to obtain a prefix allocation, namely `getAllocation` and `getSequentialAllocation`. The `getAllocation` transaction allocates a new prefix without considering previous allocations made by the InBlock6 to the requesting LIR. This is used normally when the requesting LIR has not previously received any prefix from the InBlock6. The `getSequentialAllocation` function is used to request the allocation of a new prefix that is aggregatable[12] to an InBlock6 prefix previously allocated to the requesting LIR. The `getAllocation` transaction implements the *sparse allocation* algorithm [90] to perform initial prefix allocations as separated to each other as possible. This algorithm maximizes the chances of obtaining aggregatable prefix allocations for all requesting LIRs. The sparse allocation algorithm starts by dividing the InBlock6 pool in halves, and allocates the first prefix at the beginning of the first half. The next request receives the prefix at the beginning of the second half. Then, the two halves are further divided equally, and the algorithm is applied recursively to the resulting blocks. InBlock6 checks that the prefixes selected by the algorithm have not been previously allocated before by a sequential request. If this is the case, the next prefix according to the sparse allocation algorithm is selected.

The request for sequential address allocation contiguous to a previously allocated prefix is implemented through the `getSequentialAllocation` transaction, that succeeds if the

[12]Two prefixes are aggregatable if they can be expressed as a single (larger) prefix. This can only be done if the two prefixes are subsequent one another. Aggregation is considered beneficial for the global routing system scalability.

requester already holds a prefix, and the aggregatable prefix is available.

When a LIR issues a transaction requesting a prefix, the InBlock6 smart contract allocates storage for the prefix information record, which includes the Ethereum account identifier of the LIR, the time at which the prefix has been allocated (or renewed), the ROAs, and the additional routing and policy information. Any allocated prefix needs to be renewed before the expiration date. For this purpose, InBlock6 provides the `renewAllocation` transaction.

#### 4.3.2.3. Prefix Delegation

A LIR can assign more-specific prefixes of the allocation it holds to InBlock6 End Users, by means of the `delegatePrefix` transaction. This transaction is only valid if the prefix to assign (or any part of it) has not been already assigned. A LIR can revoke a prefix assignment through the `revokeDelegatedPrefix` transaction. A LIR delegating the prefixes can revoke any assignment, as opposed to the InBlock6 which cannot revoke the allocations made to the LIRs.

#### 4.3.2.4. Recovering prefixes from expired allocations

The InBlock6 Manager discovers expired allocations with the `getIDsPrefixExpired` call. Then, through the `recoverExpiredAllocation` transaction, it inserts in the blockchain the list of prefixes discovered, so that they are eligible for next allocations.

The InBlock6 Manager is expected to asynchronously run this process for recovering expired allocations and return the resulting address space to the available address pool. Note that the expiration of allocations also serves to fix a well-known issue with storing tokens in blockchains: if the holder of the resource loses the private key that secures the allocation, the resource is lost. It is estimated that 20% of existing Bitcoins are lost because of this [89]. In the case of InBlock6, prefixes for which the holder's private key is lost are not renewed and can be returned to the InBlock6 pool.

#### 4.3.2.5. ROAs and additional information

The current holder of a prefix, an InBlock6 LIR or an InBlock6 End User, can register in the InBlock6 the AS allowed to originate a BGP advertisement for the prefix by means of a `setROA` transaction. The `setROA` function can also be used to update or delete an existing ROA. In addition, the holder of a prefix can perform a `setPolicyURI` transaction to include a link to an external Uniform Resource Identifier (URI) where contact information or routing policy for a prefix is registered. This information is expected to be defined using Routing Policy Specification Language (RPSL) [93], the standard routing policy language used in the

Internet Routing Registries. The motivation for using an URI instead of including the whole policy information itself directly in the blockchain is to reduce the cost of the transaction to update the routing policy, which depends on the amount of information stored. The information stored by the `setPolicyURI` transaction in the blockchain includes a hash of the routing policy description, proving that the policy information accessible through the URI is authentic (generated by the holder of the prefix in InBlock6), integral (unmodified), and up-to-date.

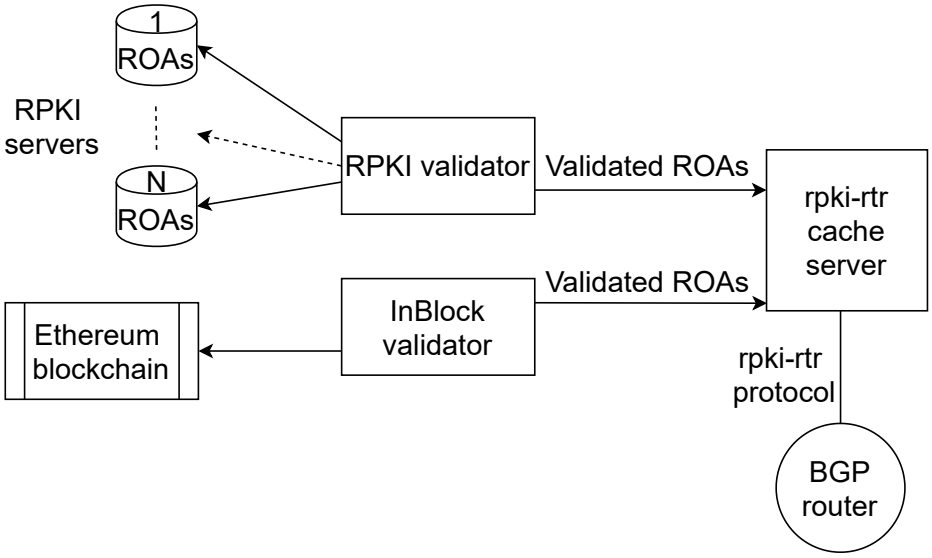


Figure 4.2: Combined validation.

#### 4.3.2.6. Third-party information retrieval

Third parties such as a router can access the blockchain to use the information stored in the InBlock6 to perform BGP Route Origin Validation (ROV). To do so, the validating router configures the InBlock6 contract identifier as a Trust Anchor for the validation of the prefixes managed by the InBlock6.

InBlock6 information retrieval can be implemented as *calls*. The `isPrefixInUse` call is used to determine if a prefix has been allocated or assigned to an End User. The get `getROA` call returns the ROA for a prefix and the `getPolicyURI` call gets the URI in which the routing policy corresponding to a prefix is stored.

The ROA information retrieved from the InBlock6 can be used to configure filter in a validating router. This can be implemented by proper modification of the RPKI to Router protocol (RPKI-RTR) (`rpki-rtr`) architecture [94]. In the `rpki-rtr` architecture routers

access through a standard protocol to a trusted cache server to acquire prefix origin data. The integration of InBlock6 into this architecture requires the modification of the software at the `rpki-rtr` server to retrieve, validate and integrate InBlock6's information into the prefix-to-origin AS state conveyed to the routers.

InBlock6 and Resource Public Key Infrastructure (RPKI) route origin validation can coexist. A validating router can use the RPKI information to validate the origin of the routes of a set of prefixes and use the InBlock6 to validate another (disjoint) set of prefixes. This can be achieved by feeding both streams of information to the `RPKI-RTR` server as shown in Figure 4.2.

To ensure that the validation information is fresh, we need to be able to update the InBlock6 state frequently. In order to efficiently support periodic updates of the InBlock6 information for all the allocated prefixes, we leverage on the blockchain monitoring capabilities, the ability of blockchain nodes to be configured to track the events associated to a particular contract and return those annotated after a given block identifier. In the case of the RPKI, RFC8182 [95] indicates that the information received by a repository must be published within a minute, but limits the polling frequency of the end users to a minute. As a result, information could be available to the routers in the order of few minutes (subject to the ability of the repositories to support requests from all validating entities at this rate). As we will show in Section 4.4.3, the blockchain monitoring capabilities can easily achieve refresh rates in the order of few minutes.

#### 4.3.2.7. Key rollover

To enable the update of the cryptographic information granting access to InBlock6 resources, InBlock6 provides functions to transfer some of the resources to different Ethereum accounts in a way which resembles key rollover procedures common in network resource management.

The InBlock6 Manager can perform a `transferInBlock6Control` transaction to transfer the ownership of the main InBlock6 contract to another Ethereum account. After executing this transaction, the holder of the new account will receive the fees resulting from allocations and renewals and it will be able to execute all the InBlock6 manager operations.

A LIR holding a prefix can also transfer its control to another account by means of the `transferAllocatedPrefixControl` transaction. InBlock6 also provides tools for the transfer of delegated prefixes. In this case, it is the LIR who indicates to which account the prefix is transferred to, through the `transferDelegatedPrefixControl` function.

#### 4.3.2.8. Emergency stop

The provision of code implementing an emergency stop is a security mechanism that allows the InBlock6 manager to block the execution of some selected functions. This provides a countermeasure to limit the damages caused by a bug or a security issue [96, 97].

In the case of InBlock6, the emergency stop function `prefixAllocationStop` suspends temporarily the `getAllocation` and `getSequentialAllocation` functions, so that InBlock6 cannot perform further allocations. The suspended functions can be reactivated by executing `prefixAllocationResume`.

This stop function does not affect the operation with prefixes already allocated. This means that previously allocated prefixes can still be renewed, delegated, ROAs can be set, etc., even if the aforementioned stop functions are executed, so that the InBlock6 manager cannot disrupt the system.

#### 4.3.2.9. Execution in light nodes

Ethereum smart contracts are executed by Ethereum *nodes*. While full nodes download and verify every block in the blockchain, it is also possible to operate with the blockchain through a *light client*, with similar security as full nodes, but much lower memory and computing requirements [98]. A light client connects to full nodes to retrieve the most recent block headers. The light client leverages in the *Merkle tree* structure to identify the blocks containing the transactions related with the InBlock6 smart contract. Then, it requests these particular blocks and validates them. Note that the Merkle tree information of the last block can be used to determine the integrity of the whole blockchain or just a stream of transactions associated to a single account. In this way, the light client can securely obtain all the information related with InBlock6, without the need to retrieve and validate the whole blockchain.

### 4.3.3. Evaluation and experiments

We next present the results of several experiments targeted to evaluate the computation/storage and monetary costs involved in InBlock6 operation as well as the delays of the different operations involved.

The implementation of InBlock6 as well as the scripts used in the experiments are available at https://github.com/steang91/InBlock_Code/tree/IPv6.

| | /20 | | | | /8 | | | |
|---|---|---|---|---|---|---|---|---|
| **Operation** | **Min Gas** | **Max Gas** | **Mean Gas** | **Mean US$** | **Min Gas** | **Max Gas** | **Mean Gas** | **Mean US$** |
| getAllocation | 209223 | 286518 | 218126 | 1.09 | 253875 | 272723 | 262512 | 1.31 |
| getSequentialAllocation | 237869 | 246119 | 242925 | 1.21 | 288196 | 296446 | 292181 | 1.46 |
| renewAllocation | 37739 | 37803 | 37786 | 0.19 | 37803 | 37867 | 37867 | 0.19 |
| recoverExpiredAllocation | 53451 | 60951 | 53482 | 0.27 | 53483 | 61015 | 53522 | 0.27 |
| getOracleCurrencyConversion | 120421 | 141568 | 130994 | 0.65 | 120421 | 141568 | 130994 | 0.65 |
| computeAllocationCost | 34703 | 34703 | 34703 | 0.17 | 34703 | 34703 | 34703 | 0.17 |
| delegatePrefix | 436466 | 1383448 | 904203 | 4.52 | 486670 | 1433652 | 954407 | 4.77 |
| setRoa | 45208 | 45272 | 45255 | 0.23 | 45272 | 45336 | 45336 | 0.23 |
| setPolicyURI | 88939 | 89003 | 88986 | 0.44 | 89003 | 89067 | 89067 | 0.44 |

Table 4.2: Gas and US $ transaction cost for different InBlock6 transactions (gas price 19 GWei, 1 Ether = US $ 263.43)

#### 4.3.3.1. Local experiments

**Goal of the experiments**: experimentally measure the computational cost of different functions implemented in InBlock6. The computational cost of executing a function in Ethereum is expressed in gas. Because the coast in Ether associated to a transaction depends both in the gas and the gas prize set for that transaction, and Ether can be converted into a fiat currency such as US$, we also express the measured cost of the different functions in US$.

**Experimental setup**: The cost in gas of executing a function solely depends on the operations included in the function and it is constant for all the nodes in Ethereum. For this reason, it is enough to measure the gas incurred for the different functions on a local node and the result will concur with the gas in any node in the real Ethereum network. Thus, to estimate the cost in gas required by InBlock6 functions, we use the blockchain emulator Ganache CLI[13] within its default configuration. Ganache CLI is part of the Truffle suite of Ethereum development tools. It offers a fast and customizable blockchain emulator. It uses `ethereumjs` to simulate full client behaviour and allows to make transactions and calls to the blockchain without the overheads of running an actual Ethereum node. The use of Ganache CLI offers the following advantages: *i)* transactions are instantly mined, *ii)* no transaction costs, *iii)* accounts can be re-cycled, reset and instantiated with a fixed amount of Ether (no need for faucets or mining), *iv)* mining speed and gas price and can be modified. The specific details of the node hardware are irrelevant, since the experiment results does not depend on it.

**Experimental methodology**: We deploy InBlock6 in the local testnet and we execute the different functions described in 4.4.2. The amount of gas required to complete a transaction depends on the associated computation and storage needs. For some functions, these needs vary with the parameters involved on each particular execution, e.g., slight differences may arise from the number of bits of the prefix that is allocated next by a `getAllocation` function. For those functions, we execute the different InBlock6 functions varying the parameters and we calculate the mean, maximum and minimum gas used.

To provide a rough estimation of the cost in US$ for each transaction, we consider the daily mean gas price offered at the Ethereum network in the three-month period from December 2018 to February 2019 [99]. The mean gas value of this series is 19 GWei[14]. For the calculation of the cost in US$ we use an exchange rate of 1 Ether = US$ 263.43.

**Experiments and results**: We perform two rounds of experiments using two different

[13] https://www.trufflesuite.com/ganache

[14] The maximum value observed is 370 GWei, reported for Feb 19th 2019; the second highest daily cost is more than ten times lower

InBlock6 root prefix lengths, /20 and /8. In both cases, the length of the allocatable prefix is a /32. The first case, /20, contains 4,096 allocatable prefixes, which we deem a reasonable size for an InBlock6 instance. The second case represents an upper bound for the size of an InBlock6 root prefix, as it covers the total amount of IPv6 addresses currently reserved for global unicast addresses (a /8 prefix). We next describe the results for each of the tested functions.

**`deploy`**: This function is executed by the InBlock6 manager once, to deploy the InBlock6 in the blockchain. The cost in gas to perform it is constant, and does not depend on InBlock6 root prefix size. We verify this by executing the function a reduced number of times, to obtain a constant result. It is not possible to deploy the whole InBlock6 contract in Ethereum through a single transaction because it would exceed the maximum gas limit per transaction, set to 6.7 MGAS. So, the code is divided into modules that are deployed separately. The total gas required for deploying InBlock6 is 12.18 MGAS, with an estimated cost of US$ 60.96. This is the most expensive operation, as it implies the storage of a large amount of bytes.

**`activateInBlock6`**: This function is executed by the InBlock6 manager to activate the InBlock6 once deployed. The cost in gas of this function is constant, 154 KGAS (US$ 0.77).

**`getSequentialAllocation`** and **`getAllocation`**: These functions are used by an InBlock6 LIR to request new prefixes, either aggregatable to a previous allocation or sparse. As different executions of these functions can require different amounts gas, we define the following sequence of transactions to evaluate their cost: execute 100 times the `getAllocation` transaction (sparse allocation algorithm), and then perform one `getSequentialAllocation`. This sequence is repeated until 4,096 prefixes are allocated. Results are presented in Table 4.4.

**`renewAllocation`**, **`delegatePrefix`**, **`setRoa`** and **`setPolicyURI`**: After running the previous test involving the prefix allocations operations, for each round of tests, the InBlock6 has 4,096 prefixes allocated. We execute the four aforementioned operations, in this case once per prefix and measure the required gas for them. Results are presented in table 4.4. The highest gas cost results from the delegation of prefixes, due to the costs of managing the tree that stores the prefix delegation information. This operation requires variable computation effort depending on the position of the prefix to allocate within the tree. The costs for managing ROAs and policies, which are expected to be the most frequent operations, are very low, well below US$ 1.

**`recoverExpiredAllocation`**. Using the state from the previous test, we artificially force to expire 100 prefix allocations and then we run `recoverExpiredAllocation`. We repeat the procedure 10 times. The results are shown in Table 4.4.

#### 4.3.3.2. Mainnet measurements

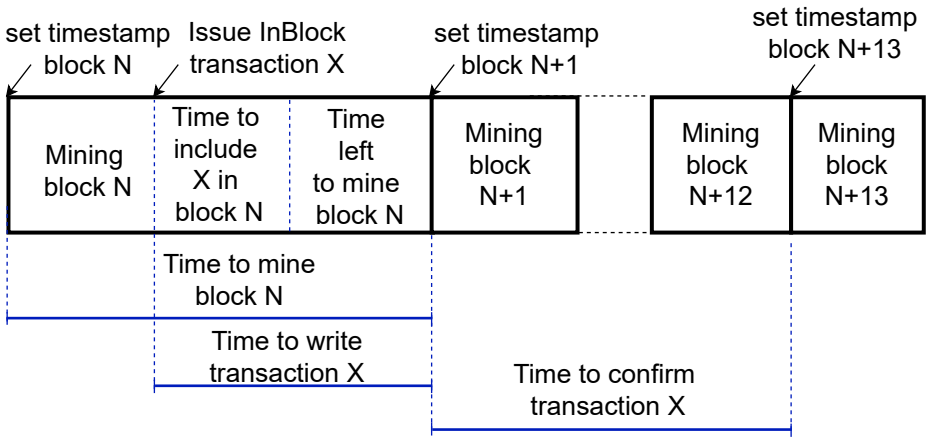


Figure 4.3: Blockchain transaction times.

**Goals of the experiment**: The primary goal of the experiment is to measure the time required for executing the different InBlock6 transactions in the Ethereum blockchain. More precisely, we measure two delays: the time to write a transaction in the blockchain and the time to confirm a transaction. The time to write a transaction is the time between the transaction is issued and the time that the transaction appears in a block in the blockchain. The time to confirm a transaction is the time to write a transaction plus the time it takes to mine additional 12 new blocks on top of the block containing the transaction [34]. The confirmation time represents the recommended time to wait to have reasonable guarantees that the transaction cannot be reverted.

A secondary goal of the experiment is to validate the results obtained in the local tests and to measure the cost of `getOracleCurencyConversion` and `computeAllocationCost` that, because they involve external third parties, cannot be measured in a local testbed.

**Experimental setup**: We deploy the InBlock6 contracts into the live Ethereum blockchain (Mainnet). To this end, we use `geth`, an implementation of an Ethereum node in the Go programming language. `geth` also serves as a console for typing commands and executing specific functions. Using `geth`, we locally setup an Ethereum light node and connect it to the Mainnet. Then, we create and fund an Externally Owned Account (EOA) account, to be able to deploy InBlock6 and interact with it to run transactions. This setup also allows us to measure both the time it takes to execute transactions in the real Ethereum blockchain, and the cost (in gas, Ether and US$) of doing so.

**Experimental methodology**: To measure the transaction write time, we issue a transaction and observe the timestamps of the new block published in the blockchain

containing our transaction. However, simply computing the difference between the timestamp of the block and the time we requested the transaction poses some issues. As illustrated in Figure 4.3, the timestamp included in a given block reflects the time when the miner started mining the block. Miners can include transactions in the block that were received after they started mining it. Thus, it is possible that our transaction appears in a block with a timestamp showing a value prior to the time the transaction was generated. In order to obtain an upper bound for the time when the block was actually mined, we use the timestamp of the subsequent block in the blockchain. This is an upper bound, under the assumption that miners start mining the next block as soon as the previous block was mined. So, our time measure is obtained by subtracting the local time when the transaction was issued from the timestamp of the block after the one where the transaction was registered.

To measure the time to confirm a transaction, the usual rule is to wait until 12 blocks are build upon the one to trust. We then measure the time elapsed until 12 additional blocks are mined and we present it along the results.

During these experiments, we also measured the gas actually used by the different transactions.

**Experiments and results**: We deploy InBlock6 on the Ethereum Mainnet blockchain, and activate it with a /20 InBlock6 root prefix, being able to allocate 4,096 prefixes. We perform 50 transactions for each of the InBlock6 functions tested and we measure their transaction write and confirmation times, as well as the gas used. Experiments were performed in March 2019. The experiment spanned over 10 hours, and each transaction is mined in a different block. We set the gas price to 19 GWei for all the experiments. In Table 4.3 we show the measured time for different transactions.

As the gas price is the same for all operations, the variations depend on the particular conditions of the blockchain when a transaction is issued. In general, we observe that the time to write a transaction is always below 3 minutes and the time to confirm an operation is below 440 seconds.

The overall time required to request a block can be estimated as the sum of `getOracleCurrencyConversion`, `computeAllocationCost` and the allocation itself (either `getAllocation`, `getSequentialAllocation`), so it is in the order of 10 minutes, as each operation should not be initiated until the previous one has been confirmed. However, the most frequent operations are expected to be policy changes, which result from the execution of individual functions, `setRoA` and `setPolicyURI`, with mean confirmation times around 3 minutes. Additionally, using these experiments we measured the gas used by the `getOracleCurencyConversion` and the `computeAllocationCost`. These functions rely on external third parties, which is why we were unable to measure them in the local tests. Results

| Operation | Mean time to write | Max time to write | Mean time to confirm | Max time to confirm |
|---|---|---|---|---|
| getAllocation | 30 | 170 | 173 | 316 |
| getSequential Allocation | 35 | 149 | 213 | 433 |
| renewAllocation | 33 | 139 | 204 | 355 |
| recoverExpired Allocation | 37 | 182 | 187 | 306 |
| getOracle CurrencyConversion | 31 | 147 | 202 | 360 |
| computeAllocationCost | 49 | 99 | 262 | 387 |
| delegatePrefix | 28 | 153 | 183 | 365 |
| setRoA | 29 | 97 | 189 | 292 |
| setPolicyURI | 25 | 84 | 186 | 293 |

Table 4.3: Time in seconds to write and to confirm different InBlock6 transactions (gas price 19 GWei)

are presented in table 4.4. The mean of the sum of the fees the Oracles request for providing their services is 0.012 Ether (US$ 3.16 according to the change rates considered). The total cost for obtaining the price of the allocation fee is the sum of the Oracle fees plus the cost of the two transactions `getOracleCurencyConversion` and `computeAllocationCost`.

## 4.4. InBlock4

We now describe InBlock4, an approach to manage address assignments that embeds a consensus-based trust model, providing alternative means to perform route origin authorisation that limits the effect of potential errors and abuses from the higher levels in the allocation hierarchy. Instead of relying on a public key infrastructure that is inherently hierarchical, InBlock4 relies on a blockchain to store allocations and route origin authorisations. The data stored in the blockchain is controlled through a consensus mechanism among the participants and cannot be modified unless there is consensus to do so. By storing prefix allocation information in the blockchain, InBlock4 embraces a trust model in which only a majority of the involved entities can revert or subvert an existing allocation.

InBlock4 is implemented as an Ethereum Distributed Autonomous Organization (DAO) for managing IP addresses. While InBlock4 can support both IPv4 and IPv6, in this work we focus on IPv4, as the majority of traffic today is IPv4. Being a DAO, InBlock4 operates completely autonomously, without any human intervention. InBlock4 provides a distributed, automatic, irrevocable, tamper-free, publicly accessible, privacy-preserving resource allocation mechanism. InBlock4 is designed to coexist with the RPKI, so that holders of resources can freely decide whether they prefer to rely on the RPKI or in the InBlock4 to provide the information needed by third parties to perform route origin validation. In this way, InBlock4 provides a non-hierarchical alternative to the RPKI while it is still compatible with BGPsec.

In this thesis, we detail the design of the InBlock4 DAO, discussing the security concerns such an implementation may face. We also perform tests to justify that InBlock4 design is able to perform the IPv4 registry functions in an efficient manner, significantly reducing the operational costs compared to a traditional registry and also reducing in orders of magnitude the time required to perform an allocation.

InBlock4 inherits InBlock6 functionalities but for the IPv4 address space, which resources are mainly already assigned [88]. Indeed InBlock provides an alternative framework to register living resources (e.g., already assigned resources) into the Ethereum blockchain, enabling the decentralised route origin validation. InBlock4 is conceived as a set of programs (smart contracts) that autonomously run in the Ethereum blockchain, performing the functions of an IPv4 address registry.

Next we describe the implementation of the InBlock4 Proof of Concept (PoC) and we show how InBlock4 behaves identifying the roles of the actors involved and we detail how basic operations proceed. Finally we evaluate its performances through an experimental analysis.

### 4.4.1. Overview Of InBlock4's Operation

Essentially, InBlock4 registers information about allocated prefixes and its holders. The holder of a prefix is identified through an Ethereum ID, which is a hash of a public key. The holder of an Ethereum ID can prove its ownership by signing with the associated private key. The holder of a prefix registered in the InBlock4 can associate additional information to the prefix in the InBlock4. In particular, it can include an explicit authorization for an AS number to announce the prefix through the global inter-domain routing system, called an IB-ROA. It can also include contact information and routing policy information about the prefix.

A third party can use the information available in the InBlock4 to perform route origin validation. By processing the information publicly available in the blockchain, a third party can retrieve the prefixes and the IB-ROAs registered in the InBlock4 and use that information to validate the origin of the routes received through Border Gateway Protocol (BGP).

The allocation of InBlock4 prefixes has a predetermined lifetime (e.g., one year). The holder of the prefix can renew the allocation simply by transferring a predetermined fee to the InBlock4 account. When an address allocation is not renewed before its expiration date, the prefix is no longer managed by the InBlock4, so the prefix returns to the Regional Internet Registry (RIR)/Resource Public Key Infrastructure (RPKI) domain. In this way, InBlock4 is able to reclaim unused addresses. Also, the holder of a prefix can transfer the prefix to another Ethereum ID, enabling an agile IPv4 address market.

InBlock4 is conceived to co-exist with currently available registries and with the RPKI. Actually, InBlock4 leverages in the information available in the RPKI to perform the initial validation of the information registered in the InBlock4. If a prefix holder wishes to register its prefix in the InBlock4, it uses the RPKI certificate as a proof of ownership of the prefix in order to register it in the InBlock4. In the current RPKI arrangement, the holder of a prefix obtained through the current RIR system can also obtain an RPKI certificate that allows it to prove the prefix ownership. If the holder of the prefix wishes to register the prefix in the InBlock4, it issues a transaction addressed to the InBlock4 contract. The transaction contains the prefix and a link to the certificate chain from the RIR root certificate to the RPKI certificate for the prefix. It also includes a hash of the certificate chain signed using the private key associated to the prefix RPKI certificate. This last signature shows that the transfer to the InBlock4 validation mechanism was requested by the rightful owner of prefix. The transaction also includes a transfer of Ether (the Ethereum cryptocurrency) to the InBlock4. This serves as a protection against Denial of Service attacks against the InBlock4, as we detail later on.

Once a prefix has been registered in the InBlock4, this information replaces the information in the RPKI (in the sense that there is no need to use the information in the RPKI to perform route origin validation for that prefix). Because the InBlock4 has a non hierarchical model, errors and abuses from higher layers in the RPKI do no affect the information in the InBlock4. InBlock4 only uses the information in the RPKI to perform the initial validation of the prefix. Any changes in the RPKI after the insertion of the prefix in the InBlock4 have no effect in the prefix, since it is now registered in the InBlock4. It would be possible for a prefix to perform the inverse operation, i.e., to cease using the InBlock4 as a registry and return to the RPKI.

When the InBlock4 receives the transaction registering a prefix, the InBlock4 does not validates that the transaction was issued by the rightful owner of the prefix (i.e., that the certificate chain is valid and that the signature included in the transaction was issued by the rightful owner of the prefix). InBlock4 simply stores the information received in the blockchain. It is up to the third parties that want to use the InBlock4 information to perform route origin validation to validate that the information stored in the InBlock4 is correct. The reason why the InBlock4 does not validates the information for the prefixes is that the operation would be expensive. Until recently, public key signature validation was not even available in the Ethereum Virtual Machine (EVM) due to its cost. Recently, it has become available, but the cost in terms of gas (processing power) still renders it unfeasible for the validation of a long chain of certificates. Thus, third parties wishing to use the InBlock4 information should verify the validity of the information stored in the blockchain by InBlock4. This process is done offline and while it may be time consuming, it only needs to be done once by the third party to be able to validate all received routes.

There is a potential risk that attackers issue a large number of invalid transactions to pollute the InBlock4 database. This would increase the work that third parties need to do in order to use the InBlock4 information. In order to mitigate this attack, the InBlock4 includes a fee that the issuer of a transaction must pay in order to add information to the InBlock4.

#### 4.4.1.1. InBlock4 Roles

The following roles with their associated operations are defined for InBlock4:

**InBlock4 Manager**. The InBlock4 Manager is the entity who executes the InBlock4 smart contract in the blockchain. Before deploying the contract, the InBlock4 manager inserts the public keys currently in use by the RIRs for the RPKI (hereafter called the *Root keys*) into the InBlock4 smart contract. The InBlock4 Manager also defines the size of the prefixes to be allocated, the size of the prefixes to be delegated and the initial fees. Then it issues a transaction to deploy the InBlock4 smart contract in the Ethereum blockchain, and activates

the contract.

**InBlock4 LIR**. An InBlock4 Local Internet Registry (LIR) is any entity that registers an IPv4 prefix in the InBlock4. The transfer of the address prefix to the InBlock4 is implemented as a blockchain transaction. Only the legitimate owner of an IPv4 prefix under the RPKI rules can transfer it to the InBlock4. The LIR includes the following information in the transaction to prove that it is the rightful owner of the transferred prefix:

- The prefix to transfer.
- A pointer to a repository (or repositories) which stores the chain of RPKI certificates from the RIR certificate to the prefix.
- A hash of the information contained in the repository, signed with the private key associated to prefix in the LIR certificate.

Once the prefix is transferred to the InBlock4, the LIR can set a IB-ROA associated to the prefix, include routing information or modify the location at which the validation information is stored. The LIR is entitled to renew periodically its prefix allocations, or remove them from InBlock4. Expired blocks do not longer belong to InBlock4, so its validation must be performed in the RIR-RPKI domain. LIRs can delegate part of the address space they hold to InBlock4 End Users, and reclaim it from them at any time.

**InBlock4 End User**. We refer to the entities which have received a prefix assignment from InBlock4 LIRs as InBlock4 End Users. InBlock4 End Users are authorized to update the InBlock4 registries associated with their assigned prefixes with contact and routing information. Part of the routing information that can be managed is the IB-ROA, to indicate the Autonomous System (AS) identifiers authorized to generate a routing advertisement for the prefix.

**InBlock4 Third Parties**. Any other entity can act as an InBlock4 Third Party by accessing to the InBlock4 information stored in the blockchain to securely retrieve information about the prefixes managed by InBlock4. In the case that a Third Party wants to obtain from the InBlock4 the information about which AS is authorized to advertise a prefix P (through the IB-ROA), it perform the steps described in Figure 4.4.

The third party validation process is as follows. The entity first checks if the InBlock4 contains authoritative information regarding prefix P. To do so, it retrieves the initial blockchain transaction related with the prefix P as well as the most recent pointer to the cryptographic repository. From the repository, it downloads the whole RPKI certificate chain from the Root certificates to the certificate containing prefix P. Then, it performs the following checks:

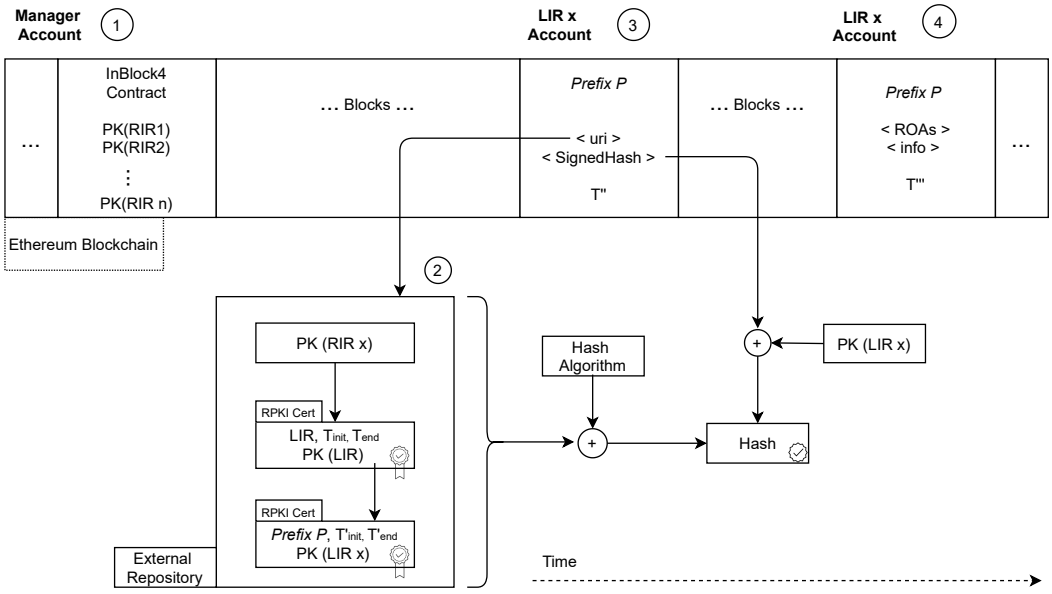


Figure 4.4: InBlock4 validation model

- The information from the repository is validated starting from the Root keys contained in the InBlock4 smart contract all the way until the certificate containing the public key associated to the prefix. The procedure to validate this information is exactly the same as defined for the RPKI information.

- It verifies that the hash included in the initial blockchain transaction corresponds to the information of the repository and that the hash was signed with the private key included at the end of the certificate chain.

- It also verifies that the prefix is active in the blockchain, i.e., it has been renewed if needed, and has not been removed from the InBlock4.

If all these checks succeed, the Third Party knows that the prefix is active in the InBlock4, and the account to which it is associated. With this 'prefix, account' tuple, it can request to the contract additional information, such as the IB-ROA.

### 4.4.2. Implementation

We have implemented InBlock4 as a set of smart contracts (a main smart contract and several utilities implemented in different contracts for convenience) written in Solidity, and deployed it in Ethereum. We next describe different relevant aspects of our implementation.

The InBlock4 source code is available at *https://github.com/steang91/InBlock_Code/tree/IPv4*.

The InBlock4 implements the different functions available to the InBlock4 manager, the InBlock4 LIR and the InBlock4 End User as Ethereum transactions. InBlock4 transactions are stored in the blockchain. InBlock4 transactions available to the InBlock4 Manager include the insertion of Root keys and the InBlock4 activation. Transactions available to the InBlock4 LIRs include the IPv4 prefix registration, IPv4 prefix delegation, and addition of IB-ROA and additional information. There is a total of 18 transactions implemented in the InBlock4 smart contract.

The functions needed for third party validation are implemented in the InBlock4 smart contract as *calls*. Call execution uses information from the blockchain, but it does not appends new information to it. Calls available to third parties include the retrieval of information related to any given prefix, such as the Ethereum ID of the holder of a prefix, the signed hash and the pointer to the repository associated to the prefix. There is a total of 9 calls implemented in the InBlock4 smart contract.

#### 4.4.2.1. Root keys Insertion

InBlock4 manager has to insert the five RIRs public keys issuing five `insertRirnamePk` transactions. Since once the activation has been done the code does not allow to changes the root keys, this operation has to be executed before performing the activation transaction.

#### 4.4.2.2. InBlock4 Activation

To activate the InBlock4 Smart Contract, which code was previously inserted in the Ethereum blockchain, a manager issues an `activateInBlock` transaction. With this transaction, the manager defines the length of the prefixes for which registration can be managed, the length of the prefix a legitimate holder can delegate and the initial fee in dollars for every registration to be inserted.

#### 4.4.2.3. IPv4 prefix registration

Any InBlock4 user can issue the `registerPrefix` transaction to the InBlock4 to register a prefix allocation and its correspondent associated information, including the signed hash and the URI pointing the RPKI repository where the certificate chain is stored. The transaction stores the inserted data in the InBlock blockchain data structure. The verification is an a-posteriori process, so the InBlock4 does not validates if the certificate chain is correct at this point.

When an InBlock4 user issues a transaction requesting a prefix insertion, the InBlock4 smart contract allocates storage for the certificate information record, which includes the ip prefix, the Ethereum account identifier of the user, the certificate id, the timestamp of the insertion, the expiration date, the ROA(s), an URI pointing the RPKI repository, an URI pointing to additional policy information, a hash of the content of the URI and a structure containing information to manage an eventual delegation such as the uri, the ROAs and the address of the delegated entity.

Any allocated prefix needs to be renewed before the expiration date. For this purpose, we provide the `renewRegistration` transaction. The internal function `isValid`, checked before every operation on the prefix, ensure that expired prefixes become invalid.

#### 4.4.2.4. IPv4 prefix delegation

Any legitimate holder can delegate a fixed part of his registered prefix issuing `delegatePrefix` function. He can even remove a delegation performing the `removeDelegation` transaction.

#### 4.4.2.5. ROAs and additional information

A statement to allow an AS to origin the advertisement of a route to that address block (i.e., a ROA) can be associated to a prefix by means of a `updatePrefixROA` transaction. The same operation is allowed for the delegated entity who has to issue a `updatePrefixDelROA` transaction.

In addition, the holder of a prefix can perform a `updatePrefixURI` transaction, or `updatePrefixDelURI` transaction for a delegated entity, to bind the prefix to a link to an external URI in which a contact information or routing policy is registered. The language in which this information is stored is the usual RPSL language [93] used in the Internet Routing Registries. The rationale for using an URI instead of including the policy information itself in the blockchain is to reduce the storage cost. While URIs and ROAs can be updated in time by the legitimate certificate holder, the hash of the original certificate must not be modified and no way to manipulate the hash is provided.

#### 4.4.2.6. Third-party information retrieval

Third parties (e.g., a validating router) can access to the blockchain to use the information stored in the InBlock4. To do so, the validating router must rely in the Ethereum blockchain and configure the InBlock4 contract identifier as a Trust Anchor for the validation of the prefixes managed by the InBlock4.

The operations for InBlock4 information retrieval do not result in modifications of the information stored in the blockchain, so they are implemented as calls.

InBlock4 information retrieval can be implemented as *calls*. Since each certificate can be retrieved issuing the key-pair (ip-prefix,id), a third party has first to retrieve the ids corresponding to the registrations associated to a prefix with `getPrefixRegistrationIds`. It can then retrieve the holder ethereum address, the uri pointing the RPKI and the signed hash executing the `getRegistrationUriRpkiSigHash` call. Once a third party has validated a prefix entry, he can issue the `getRegisteredPrefixInfo` to get the other associated information such as the uri pointing the repository where the routing policies are stored, the timestamp of the registration and the ROA of the prefix. The
`getRegistrationValidityTime` call is used to determine how long a a third-party can use the information returned by these calls to safely perform the AS origin validation of the InBlock4 associated prefix.

### 4.4.3. Evaluation and experiments

In this section we perform some experiments to measure the cost and execution time required to perform some mayor InBlock4 operations.

#### 4.4.3.1. Local experiments

We now show the *gas* and monetary cost required to complete different InBlock4 transactions. *Gas* is the unit Ethereum uses to express the cost in terms of computation and storage of a given call or transaction. A fixed amount of gas is assigned to each individual operation [100]. The cost of a transaction depends directly on the amount of gas spent on it, and also on the *gas price* offered by the requester. Increasing the gas price offer makes the request more attractive for the miners and thus reduces the time to register the transaction in the blockchain. A list of gas prices along with the estimated mining times are provided by [33], so that transaction issuers can select a price to their convenience.

To provide a rough estimation of the cost in US$ for each transaction, we consider the daily mean gas price offered at the Ethereum network in the three-month period [99] from December 2018 to February 2019. The mean value of this series is 19 GWei[15].

The experiments are executed locally in an EVM, as the gas consumed is the same in any Ethereum node. We use the amount of gas to estimate the cost in US$ of the operations.

[15]The maximum value observed is 370 GWei, reported for Feb 19th 2019; the second highest daily cost is more than ten times lower.

We show in Table 4.4 the gas required and the cost for deployment, insertion of InBlock4 Root keys, activation and prefix registration operation.

| Operation | Gas | US$ |
|---|---|---|
| deploy | 9879886 | 46.82 |
| insert root keys | 2156918 | 10.22 |
| activate InBlock4 | 48574 | 0.012 |
| register Prefix | 2164374 | 10.25 |

Table 4.4: Gas and US$ transaction cost for different InBlock4 transactions (gas price 19 GWei, 1 Ether = US$ 250)

The deployment of the Smart Contract is the most expensive operation, as it implies the storage of a large amount of bytes. Another costly operation is the insertion of InBlock4 Root keys. Five keys are claimed to be inserted before the activation can be performed. The prefix registration cost in computation is around US$ 10. Deployment, insertion of Root keys and activation are the firsts steps needed to run InBlock4 Distributed Autonomous Organization (DAO). The cost of these operations are only incurred once and are paid by the InBlock4 Manager.

We next compute experimentally the time required to retrieve validation data. Given a prefix, a third party that validates the prefix information using the InBlock4 must first retrieve from the InBlock4 the ID of the holder of the prefix, using the `getPrefixRegistrationIds` call, and also the Uniform Resource Identifier (URI) pointing to the external repository and the hash of the information stored in it, using the `getRegistrationUriRpkiSigHash` call, both calls available in the InBlock4 contract.

We donwloaded the blockchain and we measured the time required to locally execute the call to retrieve the ID associated to a prefix. The medium time required to retrieve the ID of a prefix that is only registered once in the InBlock4 is 40 ms.

Once the IDs associated to a certain prefix have been retrieved, we can get the pointer to the external repository containing the RPKI certificate chain as well as the signed hash. The times required to retrieve this information from the blockchain is between a min time of 234 ms, max time of 366 ms, and a mean time of 281 ms.

The time required by the combination of both operations is the sum of times required by both calls, i.e., a mean time of 320 ms. The combined operation is the one needed to verify which one is the valid registration and it has to be done just once per prefix. Moreover, the number of inserted prefixes does not influence this time analysis since prefix values are the keys of the map data structure used to store data.

#### 4.4.3.2. Mainnet measurements

We next analyse the time required for inserting the different InBlock4 transactions in the blockchain. For this purpose, we deploy the InBlock4 Smart Contract [101] in the Ethereum blockchain. Then, we perform 50 transactions for each of the InBlock4 functions tested, we register the time at which the transaction is requested and we obtain an upper bound of the time required to include the transaction in the blockchain.

The transaction inclusion time in the blockchain can be derived from the timestamp of its block in the blockchain. Transactions mined in the same block of the blockchain share the same timestamp. The value of the timestamp of a block must be higher than the timestamp of the timestamp of the previous block. Once the block has been mined, the timestamp should not be higher than the time measured by other miners, or the block will be rejected by them. As the timestamp can be set to the time at which the mining process of the block starts, negative values can result from subtracting the request time from the block time, specially for blocks requiring more time to be mined. Thus, to compute an upper bound of the time to complete the transaction, we use the timestamp of the block next to the one in which the transaction has been registered.

In addition, we also measure the time to confirm the transactions. As a rule of thumb, a particular transaction is considered to be confirmed if 12 or more blocks have been added to the blockchain after the block that contains the transaction. The experiment is spanned over 10 hours, and each transaction is mined in a different block. We set the gas price to 19 GWei for all the experiments.

The mean measured times for all the different InBlock4 transactions executed are:

- max time to write: 144.5 s
- medium time to write: 31.58 s
- max time to confirm: 349.12 s
- mean time to confirm: 191.23 s

In all cases, the time to confirm an operation is below 349 seconds (less than 6 minutes), with a mean value of 191 seconds (roughly 3 minutes), which is orders of magnitude lower than the operational times of a human-operated address registry.

# 5 The IRB and ASIRIA

## 5.1. Challenges

In this chapter we present Internet Routing Blockchain (IRB) and Autonomous System Internet Registry Inference for path Authorization (ASIRIA).

The IRB and ASIRIA face two main research challenges, namely:

1. Route Leaks
2. Decentralised management of the information stored in the IRR

We first discuss the nature of the challenges. Next we describe the design, the implementation and the experimental evaluation of the IRB Proof of Concept (PoC). Lately we present the design, the implementation and the experimental evaluation of ASIRIA.

### 5.1.1. Route Leaks

Over the last decade, a vast amount of effort has been devoted to secure the Border Gateway Protocol (BGP), leading to the defintion a several Border Gateway Protocol Security (BGPsec) mechanisms. Notably, the Resource Public Key Infrastructure (RPKI) [75] provides the means to perform *origin validation*, i.e., validate the Autonomous System (AS) originating a route, and BGPsec [77] enables *path validation*, so that the sequence of ASes through which a BGP route is propagated over the Internet can be verified. While these two mechanisms protect BGP against several types of common attacks (e.g., prefix hijacks), they fail to prevent another very common threat known as *route leaks*.

In June 2015, Telekom Malaysia (AS4788) leaked (i.e., announced) over 179,000 routes learned from providers and peers to one provider, Level3. The provider, preferring routes received from customers over routes received from peers, switched its forwarding path for

these destinations to AS4788, and in turn, propagated the new routes to its own peers and clients. The affected prefixes included Google, Microsoft, LinkedIn and Reddit. Preferring leaked routes resulted in severe performance degradation when attempting to communicate with these destinations [102]. This is but one of numerous route leakage incidents that occurred over the last 20 years [103], many of which resulted in major disruptions of the Internet service in large geographical areas.

A *route leak* is defined as the propagation of a route beyond its intended scope [16]. For example, as depicted in Figure 5.1, AS1 is multihomed to two providers, IPSA and ISPB. If AS1 announces the routes learned from ISPA into ISPB, this would result in a route

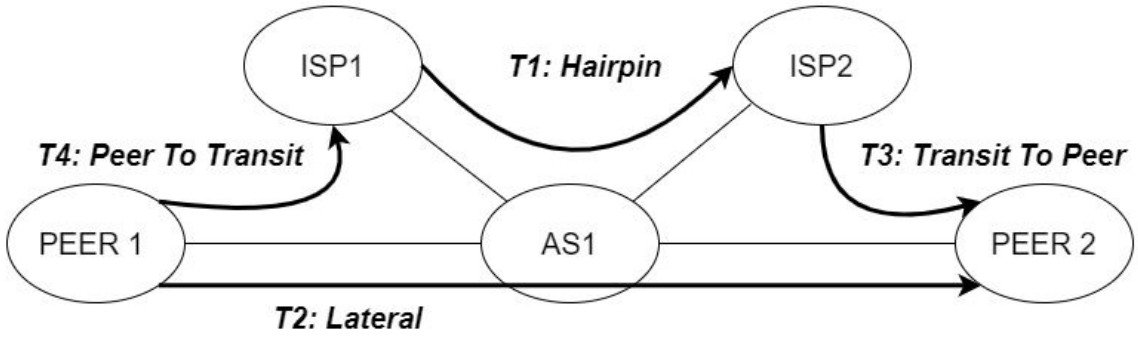


Figure 5.1: Taxonomy of route leaks.

leak since the routes announced by a provider are (in general) intended to be distributed to the customer and to the customer's clients but not to its providers. This simple example also illustrates why RPKI and BGPsec combined, fall short in protecting from route leaks. Indeed, in the aforementioned example, the routes announced by AS1 to IPSB are generated by the legitimate holder of the prefix (so RPKI/origin verification is successful) and each router along the path has propagated them through valid relationships between ASes (so the BGPsec/path validation also succeeds). As the relationship between AS1 and AS3 exists, it can be verified through BGPsec. In general, route leaks are the result of propagating routing information with a valid origin and acquired via a valid relationship beyond the scope intended by the routing policies of the involved ASes.

Internet Routing Registries (IRRs) are repositories where ASes declare their routing policies and as such, they can support route leak detection and prevention. According to the current Mutually Agreed Norms for Routing Security (MANRS) guidelines [66], operators are encouraged to use the information available in the IRR to create filters that prevent route leaks. Specifically, the MANRS guidelines on filtering recommend that operators require their direct and indirect customers to register their prefixes and AS numbers in an IRR and use this

information to create filters that discard any route announced by their customers that does not match with the registered prefixes and AS numbers. Thus, a network uses the information inserted in the IRR by the customers to generate an *accept-list* that includes all acceptable prefixes and AS-paths. The implication is that all routes that are no explicitly declared in the *accept-list* (even if they are legitimate) are filtered out. While some Internet Service Providers (ISPs) are willing to adopt such a drastic approach, many others are reluctant to do so. After all, filtering out legitimate customers routes because they have not been properly declared in the IRR may impact customer satisfaction and may have negative commercial consequences.

### 5.1.2. Decentralised management of information stored in the IRR

The Internet Routing Registry (IRR) is a globally distributed routing information database, established in 1995 with the purpose of ensuring the stability and consistency of Internet-wide routing by sharing information between network operators. IRRs are distributed repositories, individually operated by different organizations, where ASes declare their routing policies. This information can be used to perform route leak detection and prevention. According to the current Mutually Agreed Norms for Routing Security (MANRS) guidelines [66], operators are encouraged to use the information available in the IRR to create filters that prevent route leaks. In [5], we have motivated and described how ASIRIA can use the information stored in IRRs to prevent and detect route leaks serving as a bootstrap for ASPA while ASes start to register their relationship policy or their client list in RPKI servers. In short each AS registers its routing policies in the IRR. When an AS receives a route, the router of the AS uses this information, according to the ASIRIA specification, to validate it. ASIRIA takes advantage of the valley-free properties of route propagation in the internet, so that it can determine if the route is a leak or not. This, combined with origin validation, provided by the RPKI, results in quite good protection. Using the IRRs may allow to bootstrap this process and provide the ability to validate a significant number of routes even when the ASes are not yet actively registering their relationships in RPKI servers. However, there are some limitations to overcome. In this section, we will present the shortcoming of these two solution, that will serve to motivate the work on the Internet Routing Blockchain (IRB) that we present in this thesis.

#### 5.1.2.1. ASIRIA limitations

When designing and implementing ASIRIA, we identified two main limitations about using the data available in the IRR to prevent route leaks, namely, the quality of the

data and the limitations of the relationship inference algorithm. We describe them next. Regarding the quality of the information available in the IRRs, we identified two issues. First, the information may be *stalled* and second the information may be *inconsistent across the different IRRs*. Regarding stall-ness, the ASes sometimes fail to keep updated the routing policy information they registered. Regarding consistency, there are 34 IRRs and it we have observed that the information in them may diverge for the same routing object. Indeed, as each IRR runs independently, ASes may register their routing policies in different IRRs. Sometimes, this is the result of an ASes changing the reference IRR it normally uses to update their policies. When an AS starts using a new IRR, it may not remove the information in the IRR it was using previously and with time, this information may be stalled and inconsistent with the information available in the newly used IRR. We analyzed the data available in all the IRRs in March 2020 and we found that there are 67k aut-num objects, of which 5% of them are present in more than one IRR. Given that it is unlikely that one AS maintains its routing policy updated in several IRRs, these duplicated records are likely to be or become stalled. By relying on the blockchain technology, we are able to guarantee consistency across the different IRB operators by design. Regarding the limitation of the inference algorithm which goal is to determine the ASes relationship based on their routing policy, as we presented when designing ASIRIA, we are able construct inference algorithms that have a high precision. Unfortunately, we are unable to achieve 100% precision. The fundamental reason for this is that we need to infer the relationship from the routing policy declared in the aut-num objects in the IRR. While it is possible to design the IRB to support the RPSL language [93], in order to maximize the inference algorithm precision, we design the IRB so that the types of relationships are explicitly declared by the ASes in the registry. Specifically, we design the IRB to store records where an AS can declare that a relationship with another AS exists, and explicitly declare the type of this relationship. Actually p2p and p2c/c2p are by far the most common types of relationships [104]. There are other types of relationships between ASes (such as siblings, partial transit and hybrid relationships) but they are much less common, so we focus the design of the IRB in p2p and p2c/c2p types of relationships. To this end, the IRB allows ASes to explicitly declare p2p, p2c and c2p relationships. Supporting sibling relationships is trivial, because, as opposed to partial transit and hybrid relationships, siblings are defined for the whole AS. On the other hand, partial transit results in different relationships on a per prefix granularity and hybrid relationships implies that two ASes have different relationships on different interconnection points.

#### 5.1.2.2. ASPA limitations

ASPA explicitly allows ASes to declare its providers in the RPKI. As such, it addresses both limitations identified above, namely consistency of the data and difficulties in the inference of the relationships. The consistency is guaranteed because there is a single RPKI database. Relationships are explicitly declared in the ASPA record, so no inference is required. ASPA only allows the declaration of c2p relationships, but this is enough for the vast majority of the cases. The ASPA limitations are related to the governance model. As ASPA is part of the RPKI, this means that all ASPA records are under the control of the RPKI hierarchy and subject to errors, abuses and misuses, suffering from the *jurisdictional overflow* described in [1]. Furthermore several concerns have been raised about the proposal to move RPKI registry into a cloud architecture [105, 106] in order to improve performance, availability, and reduce costs. These concerns are about the loss of autonomy, independence and responsibility and the legal or political risks, and contrasts with the current IRR decentralised structure, which we argue is worthwhile preserving. Moreover in [69] has been remarked that the application of blockchain technology to the IRR system with unforgeability, distributed consensus, and provable timeline characteristics open the possibility to address many data governance problems in routing security.

Relying on blockchain technology that is inherently distributed, the IRB preserve by design the decentralised nature of the IRR, provides consistency and information stall-ness prevention and allow AS owner to store their relationship policy in a shared registry. Besides the permissioned model assure that only certified resource holders are able to store their information in the IRB.

## 5.2. Design of the Internet Routing Blockchain

In this section we describe the proposed Internet Routing Blockchain (IRB) solution. As motivated earlier, we build the IRB using blockchain technology. We decided to use a permissioned blockchain framework, more specifically Hyperledger Fabric (HF) [107].

### 5.2.1. IRB Architecture

The IRB blockchain is run by independent *organizations*, and it consist of an ordering service node and a set of network components. Every organization willing to join the IRB has to run the following HF components: a Certificate Authority (CA), at least one peer and a couchDB server. For detail regarding the HF network components we refer the reader to section 2.1.3

In [69] has been remarked that the application of blockchain technology to the IRR system with unforgeability, distributed consensus, and provable timeline characteristics open the possibility to address many data governance problems in routing security. The IRB blockchain can be set up in a customizable decentralized and permissioned fashion to address the above concerns. The IRB consortium can be started by a subset of the involved players such as RIRs, NIRs, ISPs or any interested organization. All IRB nodes maintain a copy of the shared ledger as shown in the high-level architectural view in figure 5.2.

These organisations are the ones running the blokchain. As a new transaction to register or modify a policy is performed on the system, the ledgers of every peer get synchronized. Besides instead of having a decentralized set of autonomous databases, we ensure to have an unique decentralized shared ledger to maintain updated information. In addition, there are clients of these organisations that are the ones issuing transactions that in turn populate the blockchain. For the specific case of the IRB, the model based on peers and their clients allows mapping naturally the roles of the current Internet Routing Registry (IRR), meaning that entities currently operating an IRR will become peers in the IRB while entities registering routing policies in the IRR will become clients in the IRB. This means that we expect that the peer organisations will include Regional Internet Registries (RIRs), National Internet Registry (NIRs), some large Internet Service Providers (ISPs) and other entities which purpose is solely to run an IRB instance (similar to the ones existing today running an IRB).

While the service providing ordering to blockchain transactions is in general a critical component, in this particular case it is not. The reason is that the altering the order of registration of different relationships does not have any significant impact. So, we propose that the ordering system is operated by the different peer organisations in a round robin

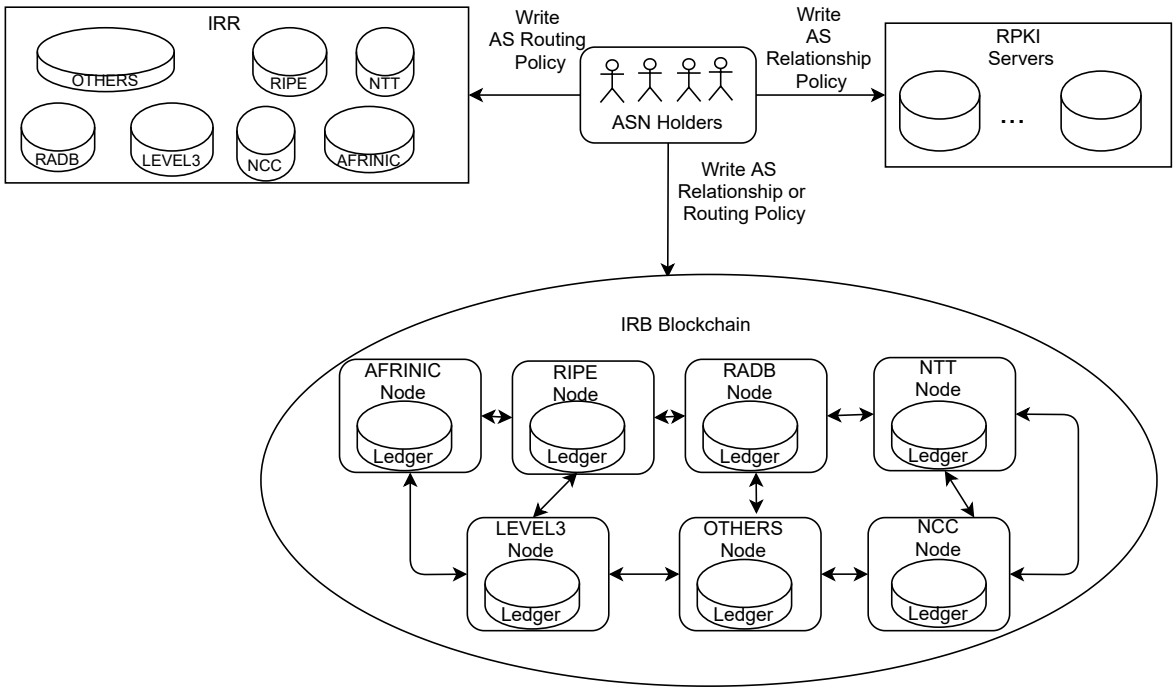


Figure 5.2: High-Level Architectural View

fashion a customizable time slot each. Whenever needed, there are already existing solution to set up a Byzantine Fault Tolerance (BFT) ordering service [108, 109] within HF.

### 5.2.2. Membership management

Being a permissioned blockchain, the IRB must manage its membership for both the organisations composing the IRB and their clients. We describe them next.

#### 5.2.2.1. Organisation membership management

Organisations are the ones running the network components of the blockchain, storing its data, validating and endorsing the transactions. The consortium, composed of the organizations bootstrapping the IRB, is in charge of managing organizations identities and membership. This means, that it is the consortium of organisations who decide to accept new peers. As example a new organization must be approved by at lest n/2 +1 of the total set of organizations to join the IRB. The specifics about how this is done is out of the scope of this work. The bootstrapping of the consortium requires an initial set of organisations to agree to build the IRB, for which a good candidate set would be a subset of the RIRs as it has been simulated in this work.

#### 5.2.2.2. Client membership management

Any entity holding an Autonomous System (AS) number can become a client of the IRB. Only legitimate holders of the AS number resources can register relationships involving this AS number. This naturally implies that the IRB membership is tightly related to ownership of AS numbers. We consider two different situations of an entity holding an AS number. The first situation is when the ASN holder has an Resource Public Key Infrastructure (RPKI) certificate associated to the ASN [110]. In this case, the ASN holder can prove that it is the legitimate holder of the ASN by exhibiting control of the private key associated to the public key contained in the certificate. So, any entity having an RPKI certificate should automatically be accepted as a client in the IRB and be allowed to issue transactions affecting that ASN. In particular, each entity having an RPKI certificate attesting the ownership of an Autonomous System Number (ASN) issues become a client of the peer organization ran by the RIR who issued the said certificate. The second situation is when the holder of the ASN has obtained it from a RIR/NIRbut does not have an RPKI certificate. In this case, the entity is client of the RIR that allocated the ASN and it is the RIR that validates that the allocation is legitimate using its own means. In this case, the client will obtain a certificate from the peer organisation certification authority to be able to issue transactions in the IRB.

### 5.2.3. Asset description

IRB Blockchain network participants perform *ASN-pair contract transactions* to achieve their business objectives such as the registration of the business relationship between two ASes. The ASN-pair contract defines the ASN_PAIR asset, an object characterized by a list of states, namely *Issued*, *Verified*, *Invalid*, and of a set of transactions that trigger events and state transitions such as *Issue*, *Sign* and *Invalid*. A security level is assigned to each state of the asset. The security level is the number of recognized interaction between certified ASN holders on an specific mutual relationship. In detail securityLevel=1 is associated to *Issued* state, securityLevel=2 to *Verified* state and securityLevel=0 to *Invalid* state. The ASN_PAIR asset life cycle is defined by the finite state machine in figure 5.3.

The ASN_PAIR record is finally described by its set of fields:

- State
- Autonomous System Number 1
- Autonomous System Number 2
- Relationship

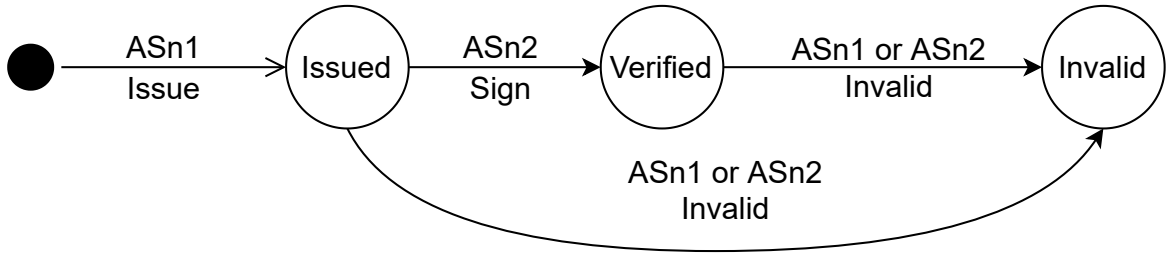


Figure 5.3: Life Cycle of Holdership Attestation

- securityLevel

Each ASN_PAIR record is accessible by the primary keys [ASN1,ASN2] and [ASN2,ASN1].

We now describe a sample scenario, depicted in figure 5.4, where the holder of ASN1 declares the business relationship established with the holder of the ASN2 using the IRB blockchain.

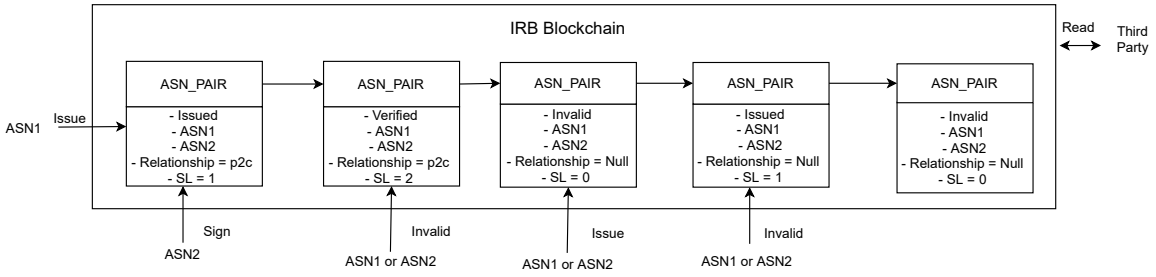


Figure 5.4: Interaction with Blockchain

The ASN1 holder performs an initial *Issue* transaction. The contract checks the existence of an ASN_PAIR record with the same keys [ASN1,ASN2] and [ASN2,ASN1]. If keys do not match with an existing record a new ASN_PAIR record is created. Its securityLevel is set to 1 and its state to *Issued*; If keys match an existing record, the contract allow its modification only if its state is set to *Invalid*. Otherwise the contract raises an error. Once the ASN_PAIR with keys [ASN1,ASN2] has been issued, the ASN1 holder can only invalid the record performing an *Invalid* transaction which sets the record security level to 0 and state to *Invalid*. Meanwhile the ASN2 holder has the right to perform or an *Invalid* transaction or a *Sign* transaction. The *Sign* transaction sets the ASN_PAIR record security level to 2 and state to *Verified*, in the means that has been publicly recognized that both parties involved in the business relationship have officially signed the described relationship. At this stage, whether ASN1 or ASN2 holder wants to modify the relationship, he has to invalid first the previous record and then re-issue the record with the modified information. Hence this record

has to be signed again by the counterparty to increase its security level to 2.

Even if an information with an higher security level is always preferred, to perform leak prevention it is possible to use an ASN_PAIR record which security level is 1. In the specific case where an AS agrees with the declarations of its neighbors, it may not need to sign its record. However, this mechanism provides means to easily invalidate the declarations of other party.

### 5.2.4. Third Party Data Retrieval

A third party who wants to retrieve the information stored in IRB can accomplish the task in two different manners: contract call or via simple http requests. The maximum number of records that can be requested in one contract call is limited to 100 thousand for security reason. In case there is the need to retrieve n*100 thousand records at same time, it is possible to use pagination [111] to split the n request or to perform a simple http request. We designed the IRB to query data via http requests which in turn is faster and has no limitation regarding the number of records to be requested in a single operation. Besides an interested third party can sing an event subscription to the IRB to receive all the event generated by the contract triggered by transactions. Using an event based model to manage contract transactions provides an easy traceability of the incremental differences generated by the newly transaction insertion.

## 5.3. Evaluation and experiments

In this section we the evalution of the interactive prototype of IRB blockchain [112]. Tests to show that the contract is functionally correct are also available at [112].

### 5.3.1. Dataset and experiments

We experimentally compute the data retrieval time and the amount of memory required as the number of relationships registered grow.

We first describe the dataset that has been used for testing the IRB blockchain's scalability. The dataset contains information regarding the AS relationships inferred by Cooperative Association for Internet Data Analysis (CAIDA) [113], which represents a subset of the Internet's relationship, combined with the Internet Assigned Number Authority (IANA)'s data regarding the ASN assignments to RIRs [114], in figure 5.5. The resulting dataset is the information regarding the AS relationship depicted per RIR. As this information is divided in 5 organizations and it comes from data that are mainly considered

as a Ground truth, we can simulate a close-to-real scenario for testing the IRB blockchain as a consortium run by RIRs. The total number of unique ASN counted is about 72K and

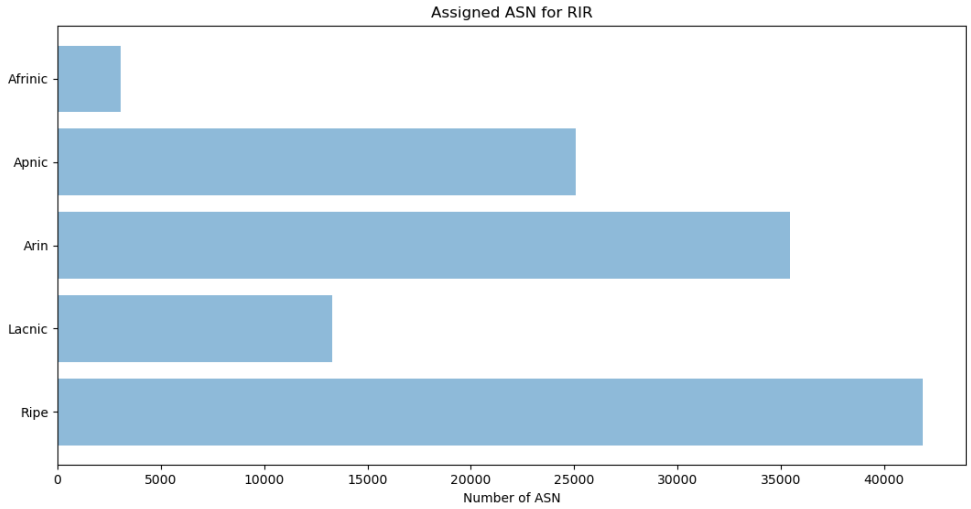


Figure 5.5: ASN assignments

it represents the total number of users as clients that can participate in the IRB blockchain at the time of this writing. The number of new relationships in figure 5.6, depicted per RIR for year, is the amount of information that has to be inserted to bootstrap the system with the current available information. The number of changed relationship, depicted per RIR for year in figure 5.7, is the amount of information that has suffered changes from year to year and changes are an estimation of the modifications that has to be registered. The total amount of records to be inserted to simulate the actual state of ASes information, based on CAIDA and IANA data, is about 915 K of records. As each new record requires 200 Bytes, this result in at least 200 Megabytes of data. As expected, the time of information retrieval and the couchDB memory size are directly proportional to the number of records and they increase linearly. Respectively the time to retrieve data is comprised between 0.1 and 14 seconds while the the CouchDB memory size between 0 and 200 Megabytes.

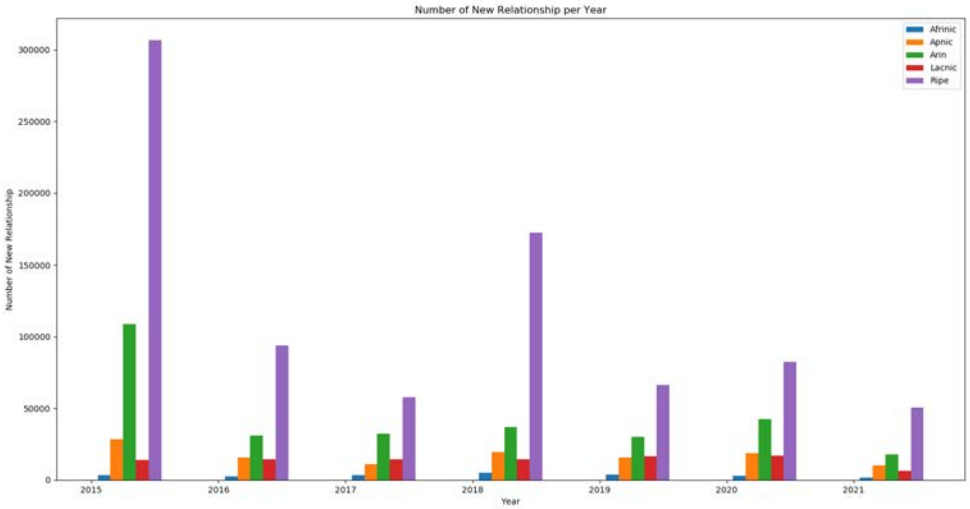


Figure 5.6: New relationships per year

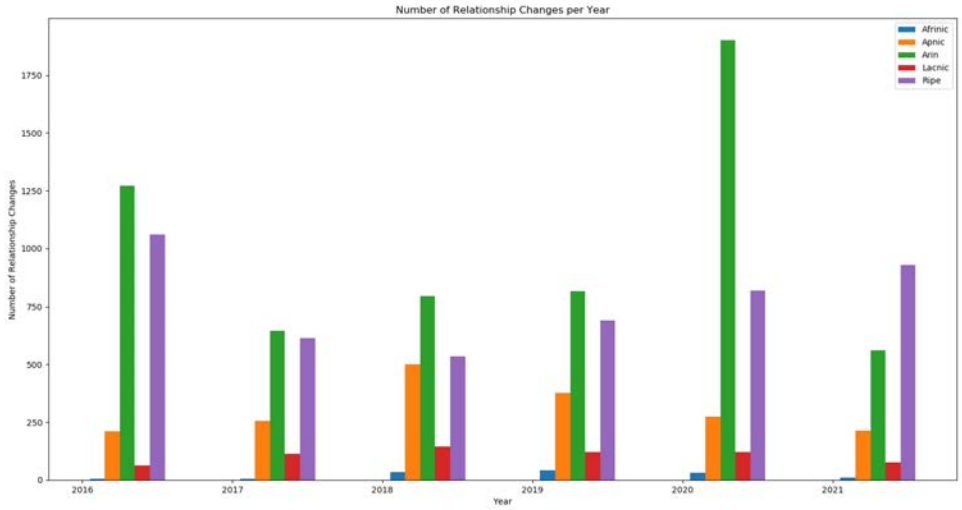


Figure 5.7: Relationship changes per year

## 5.4. Design of ASIRIA

In this work, we propose a shift in the paradigm on the use of Internet Routing Registry (IRR) information to prevent leaks, from the current *accept-list* one to an alternative one based on *drop-lists*. We introduce Autonomous System Internet Registry Inference for path Authorization (ASIRIA), a novel mechanism to provide protection against route leaks based on *drop-lists*. ASIRIA infers the type of relationships between Autonomous System Numbers (ASes) from the information available in the IRR. Using the information about ASes relationships, ASIRIA identifies and avoid invalid routes i.e. leaks. Invalid paths, are detected by analyzing the relationships between consecutive AS present in the AS_PATH attribute of a Border Gateway Protocol (BGP) route. Valid routes contain *valley-free* paths[1] [115] while invalid routes/leaks have AS paths that are not *valley free* [16]. ASIRIA identifies routes containing non *valley free* paths and include them in the *drop list*, in order to avoid using them if alternatives routes are available. ASIRIA also keeps track of the number of leaks detected and uses this information to build leakage event alarm mechanism, that allows ASIRIA-enabled routers to detect leakage events and notify the network administrator of suspicious behaviour.

By using a *drop-list* instead of an *accept-list*, operators do not have to filter out legitimate customer routes even if the customers missed to insert their information in the IRR, avoiding frictions with customers. However, the effectiveness of the *drop-list* depends on the quantity and the quality of the information available in the IRR and mobilising a large fraction of the ASes in the Internet to populate the IRR is indeed challenging. However, thanks to the long lasting efforts of the Internet operational community in promoting the population of the IRR, a significant number of ASes have already registered routing policy information in the IRR. We show that, using current IRR information, we can infer 97k AS relationships with a precision of 96,7%. Using this information already available in the IRR, we compute the performance of ASIRIA using real routing information from 303 ASes and we find that, in the initial phase, the ASes implementing ASIRIA can detect and reject, in average, 17% of leaked paths in case a customer or a peer leaks the full BGP routing table and that the top 20% of the measured ASes can detect at least 40% of leaked paths. Regarding the detection of leakage events, 90% of the analyzed ASes can detect more than 99% of the leakage events simulated in our analysis. We believe that these results show that ASes implementing ASIRIA can obtain immediate benefits leveraging on existing IRR information and that this can drive

[1] Valley-free paths have a pattern in which packets go from a customer to a provider (possibly many times), then may go to a peer, and descend to a customer (possibly may times) until the destination is reached. In this pattern, every AS in the path has an incentive for forwarding the traffic. Subsets of this pattern are also reasonable from the economic perspective. Non valley free paths are those with any other pattern.

ASIRIA adoption, fostering other ASes to populate the IRR, enabling a virtuous cycle of IRR population and ASIRIA adoption.

### 5.4.1. Route leaks classification.

The most common relationships established between two ASes are Provider to Customer (P2C) and Peer to Peer (P2P) [104]. In the former, the provider agrees to propagate all its routes to the customer, while the customer injects its own routes and those from its customers into the provider. The intention of this settlement is to allow the customer to access to the Internet through its provider. In a P2P relationship, both participants exchange their own routes and those from their customers, enabling shortcuts between the involved ASes. In this way, they avoid transiting through their respective providers. There are other relationships, such as siblings, partial transit, but they are much less common than the ones described above.

RFC 7908 [16] defines 4 types of route leaks that are characterised by the order in which P2P and/or P2C relationships appear in the AS path attribute of the route announced through BGP. As illustrated in Figure 5.1, the four types of route leaks are the following:

- Type 1 - Hairpin: A (multihomed) client announces a route learned from one provider to another provider. The AS path of the leaked route contains a P2C relationship, followed (not necessarily back to back) by a c2p relationship.

- Type 2 - Lateral: An Internet Service Provider (ISP) announces a route learned from peer to another peer. The AS path of the leaked route contains (at least) two P2P relationships.

- Type 3 - Transit to peer: An AS announces routes learned from a provider to a peer. The AS path of the leaked route contains a P2C relationship followed (not necessarily back to back) by a P2P relationship.

- Type 4 - Peer to transit: An AS announces routes learned from a peer to a provider. The AS path of the leaked route contains a P2P relationship followed (not necessarily back to back) by a Customer to Provider (C2P) relationship.

The type and order of relationships present in each of the types of leaks are presented in Table 5.1. We are able to detect route leaks by identifying these patterns. Note that all leaks are generated when a route is advertised either to a provider or a peer, but never when a route is advertised to a customer.

Additionally RFC 7908 defines route leaks of type 5 and 6, but these leaks are related to mis-origination or re-origination of routes by the offending AS, so they are suited to be

| Leak Type | AS relation sequence |
|---|---|
| 1 | ...p2c...c2p... |
| 2 | ...p2p...p2p... |
| 3 | ...p2c...p2p... |
| 4 | ...p2p...c2p... |

Table 5.1: Route leak types

addressed by existing techniques such as Resource Public Key Infrastructure (RPKI)/Route Origin Authorisation (ROA)-based origin validation. For that reason, in the rest of this work, we only consider route leaks of types 1 to 4.

### 5.4.2. The ASIRIA mechanism for route-leak mitigation

ASIRIA is a mechanism for route leak detection and protection. ASIRIA has three main components, namely, an algorithm to infer relationships between ASes from the information available in the IRR, a mechanism running in the ASIRIA enabled routers that uses the inferred AS relationship information to perform real time detection of invalid routes, and an alarm system that detects leakage events.

The ASIRIA inference algorithm extracts C2P and P2P relationships from the IRR using the `import` and `export` family of attributes present in the `aut-num` objects. In Section 5.4.3.1, we detail the process to infer each type of relationship and we also show that by analysing the information registered in the IRRs by the ASes themselves, we are able to determine with high confidence a significant number of relationships between neighboring ASes.

The ASIRIA inference process is performed off-line by the ASIRIA Local Server (ASIRIA LS) that mines the IRR database to extract the relationship between neighboring ASes, as depicted in Figure 5.8,

The information obtained contains entries in the following form:

(AS1, C2P, AS2), indicating that AS1 is a customer of AS2.

(AS1, P2P, AS2), indicating that AS1 and AS2 have a peering relationship.

This information is conveyed to an ASIRIA-enabled BGP router. The mechanism to transfer the information from the ASIRIA LS to the router is similar to the RPKI-RTR protocol [116], by which the information originated in the RPKI is installed in a router. The ASIRIA information is stored at the router in the *ASIRIA Data Base* (ASIRIA DB). In addition, the network manager should manually feed information about the relationship with neighbouring ASes, if such relationships cannot be inferred from the IRR data.

Using the information available in the ASIRIA DB, an ASIRIA-enabled router executes

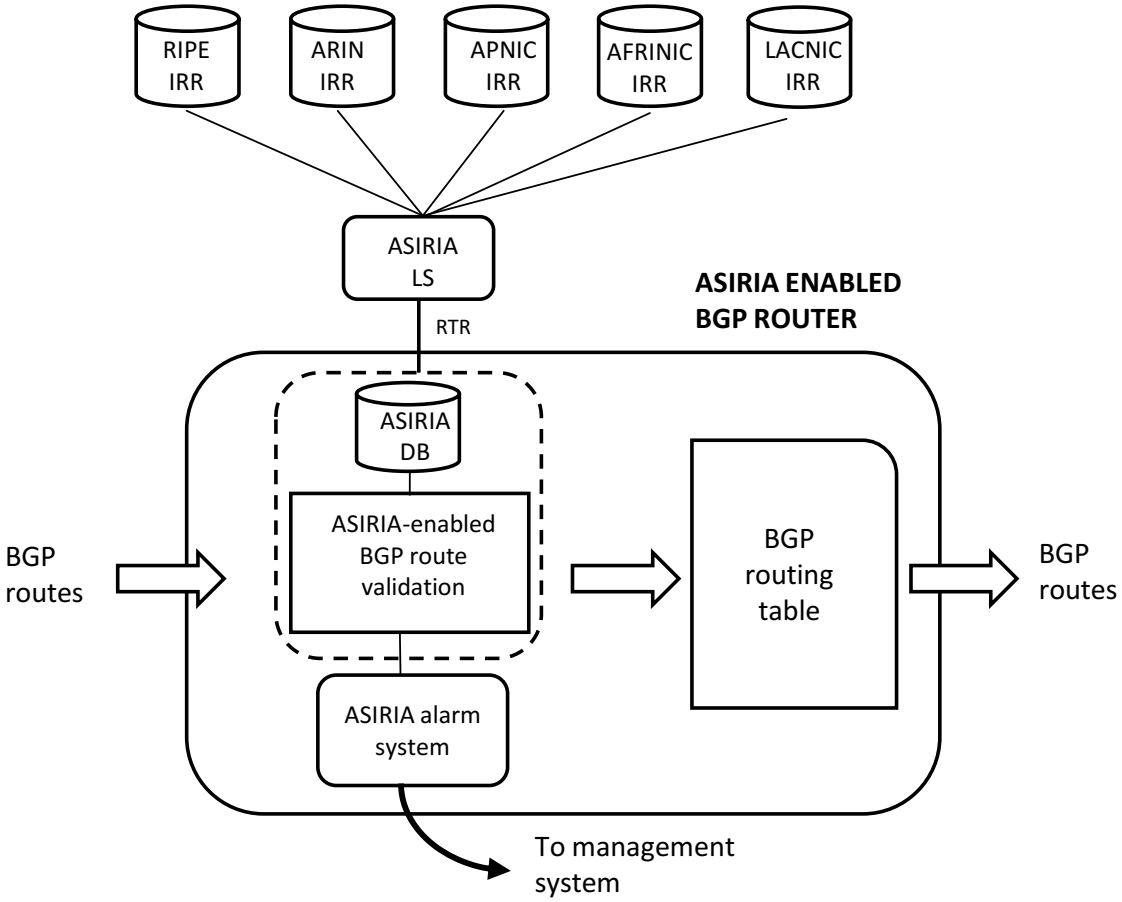


Figure 5.8: ASIRIA architecture

the ASIRIA route validation process. When it receives a BGP route, it extracts the pairs of ASes included consecutively in the AS_PATH attribute of the route and searches for matches in the ASIRIA DB. It then searches for patterns of AS relationships present in the AS path that would imply that the route is a leak, i.e., those in Table 5.1. Such routes are deemed invalid. In many cases, it is possible that the relationship for one or several of the AS pairs present in the AS path is unknown. These AS pairs are simply ignored, and the validation process continues to look for other indications that the route may be invalid.

If a route is marked as invalid, the router decreases its priority, so that other routes for the same destination, not marked as invalid, are preferred. The reason for not plainly discarding invalid routes is that the information on the IRR may be wrong (e.g., outdated) or incomplete, causing the ASIRIA algorithm to make mistake when inferring a relationship. By

de-prioritising invalid routes, ASIRIA is able to select alternative valid routes if they exist but still use routes marked as invalid if no alternative route is available. So, by design, ASIRIA assumes that there may be errors in the inferred relationships and it treats the invalid routes accordingly, limiting the effects of such errors. This design choice allows ASIRIA to work with inaccurate information.

In addition to de-prioritise specific invalid routes, ASIRIA also detects leakage events and triggers an alarm that can serve as an early warning to the system administrator. The ASIRIA alarm system works as follows: ASIRIA maintains statistics about the number of preferred routes that are marked as invalid. In a normal operation, i.e., when there is no ongoing leak, this number is expected to be small, accounting for errors of the AS relationship inference mechanism and valley paths. The ASIRIA leakage event is triggered when the number of preferred routes that are marked as a leak suddenly increases beyond a configured threshold, i.e., a 100% increase in a 5 minute period.

### 5.4.3. Description and validation of AS pair relationship inference

#### 5.4.3.1. Criteria to infer relationships between ASes using IRR data

ASIRIA uses the data available in the IRR to infer relationships between ASes. To do so, it relies on the information included by the ASes in their respective `aut-num` objects, in particular in the `import` and `export` attributes. The `aut-num` class is defined in [93] and it is used by an AS to specify its routing policy. The `import` and `export` attributes specify the import and export policies respectively.

As described in [117], in case of a transit relationship in which AS1 is a provider of AS2, the common configuration for the `import` and `export` attributes are:

```
t1) aut-num:  AS1; import:  from AS2 accept AS2 + AS2 customer set
t2) aut-num:  AS2; export:  to AS2 announce ANY
t3) aut-num:  AS2; import:  from AS1 accept ANY
t3) aut-num:  AS2; export:  to AS1 announce AS2 + AS customer set
```

Through this configuration, AS1 states that it exports its full routing table while it only accepts routes originated by its customers and its customers' customers. AS2 states that it accepts all the routes announced by AS1, while only exports a subset of the routes present in its routing table, typically its own and those belonging to its customers. We also account for alternative forms to express the same routing policy using the `mp-import`, `mp-export`, `default`, `mp-default` attributes. To simplify the exposition, and without loss of generality, for the rest of the description of the policies, we solely use the `import` and `export` expressions, which are the most commonly used.

In the case that two neighbouring ASes have a P2P relationship, this is represented in the IRR as follows:

```
p1) aut-num:  AS1; import:  from AS2 accept AS2 + AS2 customer set
p2) aut-num:  AS1; export:  to AS2 announce AS1 + AS1 customer set
p3) aut-num:  AS2; import:  from AS1 accept AS1 + AS1 customer set
p4) aut-num:  AS2; export:  to AS1 announce AS2 + AS customer set
```

Through this configuration, both ASes express that they accept and announce prefixes belonging to the peering ASes and to their respective customer's cones.

In the case of a sibling relationship (i.e., when two ASes provide mutual traffic), this is represented in the IRR by both ASes declaring both import ANY and export ANY rules. However, a sibling relationship has no effect when detecting a leak i.e. its presence or absence has no impact in the leak detection algorithm. Because of this, ASIRIA does not specifically infers sibling relationships even though it would be possible to do so from the IRR data.

There are other more sophisticated relationships, such as *partial transit*, in which the provider offers transit to a subset of Internet destinations, and *hybrid relationships* (in which two ASes have different types of relationships in different interconnection points) [113]. It is possible to describe these relationships using the `aut-num` objects above, but because they are less common (they account for 4.5% of the relationships according to [113]), we focus on inferring the P2P, P2C and C2P relationships that account for the vast majority of relationships.

#### 5.4.3.2. Inferring relationships out of the IRR data

We next extract information about AS relationships from the IRRs. There are 25 IRRs according to IRR.NET. It is documented that the quality and the amount of data in different registries varies heavily [118], some of them containing stalled information and being inconsistent with each each other. For this work, we use the IRRs maintained by the RIRs, i.e., RIPE, APNIC, ARIN, AFRINIC and LACNIC, as they provide strong authentication means, guaranteeing that only the legitimate owner of the AS can update the information associated to its `aut-num` object.

We retrieve the IRR databases on the 2021-03-01. We find that most of the information comes from RIPE database. In particular, it contains 644,039 policy expressions (`import`, `export`, `mp-import`, `mp-export`, `default` or `mp-default`), while the rest of the IRRs account for just 39,139 expressions. We filter out also those `import` and `export` attributes that contain the `refine` and `except` keywords, since they are very few of them and they are hard to parse (because of this, we filtered 650 objects out of the 600k expressions reported above). We

expand the `as-set` attributes present within the retrieved `aut-num` objects.

The IRR data is frequently incomplete e.g. sometimes only one of the ASes involved in a relationships declares it in the IRR, some ASes declare either the `import` or the `export` but not both. So we expect that it will be fairly frequent that not all the four expressions are available for many pairs of ASes. So, we search for the pairs of ASes that match a subset of the conditions described earlier characterising the C2P and the P2P relationships.

An additional difficulty that we encounter when inferring relationships from the IRR data is that several of the expressions contained in the `aut-num` objects describing both C2P and P2P relationships refer to the customers of the respective ASes, which we have not identified yet. So, we start by identifying candidate sets of C2P relationships and P2P relationships, by using conditions that do not assume any knowledge of inter AS relationships and we refine the inferred relationships in a later stage. The conditions for creating the candidate set for C2P relationships are:

```
tc1) aut-num:  AS1; import:  from AS2 accept filter != ANY
tc2) aut-num:  AS1; export:  to AS2 announce ANY
tc3) aut-num:  AS2; import:  from AS1 accept ANY
tc4) aut-num:  AS2; export:  to AS1 announce filter != ANY
```

The conditions for creating the candidate set for P2P relationships are:

```
pc1) aut-num:  AS1; import:  from AS2 accept filter != ANY
pc2) aut-num:  AS1; export:  to AS2 announce filter != ANY
pc3) aut-num:  AS2; import:  from AS1 accept filter != ANY
pc4) aut-num:  AS2; export:  to AS1 announce filter != ANY
```

In all these expressions `filter != ANY` means that the expression exists and that the filter registered is different than ANY. Using these expression, we create a candidate set for both C2P and P2P relationships.

We validate the relationships in the candidate sets (both for P2P and for C2P relationships) using the information about AS relationships from the Cooperative Association for Internet Data Analysis (CAIDA) AS relationship dataset[2] [119]. According to [119], the dataset is accurate to 99.6% when inferring C2P relationships and 98.7% for P2P relationships.

With respect to C2P relationships, we find that there are 27,051 AS pairs that fulfil

[2]In order to validate the algorithm presented in [119] for generating the CAIDA AS relationship data set, the authors used 3 datasets, direct reports from network operators, IRR data and information from BGP communities. The use of IRR data for validation creates a potential circular dependency. However, we observe that out of a total of 50,504 relationships they used for validation, only 6,530 were obtained from the IRR, while the remaining 43,974 were obtained from other sources.

| Conditions | IRR | Matching with CAIDA | Non-matching p2p | Non-matching p2c |
|---|---|---|---|---|
| t1t2t3t4 | 25,196 | 16,202 / 99.2% | 121 / 0.7% | 16 / 0.1% |
| t1t2t3 | 170 | 93 / 97.9% | 1 / 1.1% | 1 / 1.1% |
| t1t3t4 | 1,264 | 910 / 97.6% | 22 / 2.4% | 0 / 0.0% |
| t2t3t4 | 1,074 | 612 / 99.7% | 1 / 0.2% | 1 / 0.2% |
| t1t2 | 23,556 | 6,481 / 96.7% | 202 / 3.0% | 21 / 0.3% |
| t2t3 | 26 | 10 / 100.0% | 0 / 0.0% | 0 / 0.0% |
| TOTAL | 51,286 | 24,308 | 347 | 39 |

Table 5.2: Number of inferred C2P relations and validation using CAIDA information. For each set of conditions detailed in the first column, the second column contains the total number of relations inferred from the IRR. The third column contains those relationships that are present in the CAIDA dataset and coincide with the relationship assigned by CAIDA. In the fourth column, the relationships inferred from the IRR as a P2C and by CAIDA as P2P, and the last column shows the relationships inferred from the IRR as P2C, and by CAIDA as C2P (in the opposite direction).

the 4 conditions (*tc1*, *tc2*, *tc3* and *tc4*) and that among those also present in the CAIDA dataset, 98.9% match the type of relationship in both datasets, so we conclude that these are suitable candidates for customer provider relationships. For the AS pairs that match only 3 of the conditions (because the fourth one is not present in the IRR) that are present in both datasets, they all have more than 95% match with the CAIDA dataset, making them also suitable candidates for C2P relationships. Among the AS pairs that match only two conditions, we observe that the results are not uniform. For AS pairs fulfilling *tc1 tc2*, *tc1 tc3* and *tc2 tc3* that are also present in the CAIDA dataset, the level of matching is higher than 95%, so they are also suitable candidates for C2P relationships. For the other pairs of conditions, the matching with the CAIDA dataset decreases and it is lower than 90%. This is expected as, for instance, the conditions *tc1 tc4* are not specific for C2P relationships (note that P2P relationships use the exact same expressions). We exclude these from the P2C candidate set. So, we keep the relationships inferred using the four *tc* conditions, all the combinations of three of the *tc* conditions and *tc1 tc2*, *tc1 tc3* and *tc2 tc3* as candidate set for c2p relationships.

With respect to the P2P relationships, we find that the relationships inferred using the four *pc* conditions, as well as the ones inferred using *pc1 pc2 pc4* and *pc1 pc2* have a matching with the CAIDA dataset higher than 95%, so we keep those as the candidate set for the P2P relationships. Other combinations have a matching lower than 90%, so we exclude them from the candidate set. We looked deeper into the set of relationships that matched the set of conditions *pc1 pc2 pc3*, since excluding it seems to be at odds with including the AS pairs that comply with *pc1 pc2*. We find that there is a very small set of AS pairs that comply

| Conditions | IRR | Matching with CAIDA P2P | Non-matching p2c | Non-matching c2p |
|---|---|---|---|---|
| p1p2p3p4 | 4,003 | 548 / 97.7% | 6 / 1.1% | 7 / 1.2% |
| p1p2p4 | 1,460 | 297 / 95.8% | 0 / 0.0% | 13 / 4.2% |
| p1p2 | 41,136 | 4,937 / 95.1% | 86 / 1.7% | 168 / 3.2% |
| TOTAL | 46,599 | 5,782 | 347 | 92 |

Table 5.3: Number of inferred P2P relations and validation using CAIDA information. For each set of conditions detailed in the first column, the second column contains the total number of relations inferred from the IRR, the third column contains the number of those that are present in the CAIDA dataset and match with the relationship resulting from CAIDA's inference process. In the fourth column are the relationships that appear in CAIDA, but do not match, with CAIDA showing a P2C relationship (AS1 provider of AS2). The fifth column represents the relationships appearing in CAIDA, with CAIDA showing a C2P relationship (AS2 is a provider of AS1).

with *pc1 pc2 pc3* (only 243 AS pairs) and out of these, only 45 of them are also present in CAIDA, 40 of them match with the CAIDA and 4 of them do not. We inspected closely those not matching with CAIDA and we find that the mis-inference of the relationships were generated by a single (large) AS whose policies referred on an undefined AS-set. So, it is likely that AS pairs complying with *pc1 pc2 pc3* have indeed a P2P relationship, but given the small number of relationships inferred, a single error in a large AS causes a mismatch that results in excluding them from the analysis. In any case, given that they are very few relationships, we expect the impact to be negligible.

Once we have the initial candidate sets for both P2C and P2P relationships, we use the information provided by these candidate sets to remove the relationships that refer to a peer or to a customer in conditions *tc1*, *tc4*, *pc1*, *pc2*, *pc3* and *pc4*. Once we eliminate these from the candidate sets, we contrast against CAIDA and verify the resulting set of P2C and P2P relationships. In tables 5.2 and 5.3 we include the final number of P2C and P2P relationships inferred respectively as well as their matching with the CAIDA dataset. We infer a total 51,286 customer provider relationships and 46,599 peer to peer relationships.

### 5.4.4. ASIRIA route verification process and its integration with ASPA

We adapt the ASIRIA route verification to be a plug-in into the Autonomous System Provider Authorization (ASPA) route verification process. For details about ASPA we refer the reader to section 3.2.1.

When a router receives a BGP route, it starts by executing the route verification process defined for ASPA. It extracts the pairs of ASes included sequentially in the AS_PATH attribute of the route. To detect route leaks, the router looks for registered relationships for

the AS pairs at the AS_PATH. It first searches for a match in the ASIRIA DB and only if the relationship is Unknown by ASIRIA, it searches for a match with the information available in the ASIRIA DB. This means that the information in the ASPA DB takes precedence over the information available in the ASIRIA DB. The semantics of the information included in the ASIRIA DB is not exactly the same as for the ASPA DB. The main difference is that when an AS issues an ASPA record, it indicates all the providers of the AS, and thus, allows ASPA to conclude that all ASes that are not included in the AS' ASPA record are not its providers. This means that once an AS issues an ASPA record, all providers must be enumerated. In the case of ASIRIA, ASIRIA is capable identify C2P/P2C relationships with a high degree of confidence but the lack of information about the relationship between a pair of ASes in the ASIRIA DB does not necessarily means that there is no relationship between the pair of ASes. This results in an sligthly different validation process for ASPA and ASIRIA.

In order to combine ASPA and ASIRIA information to perform route-leak detection, ASIRIA introduces the following modifications to the ASPA validation process: Whenever the outcome of the Customer-Provider Verification Procedure for (AS1, AS2, AFI) over the ASPA DB returns Unknown, a verification procedure for the same input data over the ASIRIA DB is issued. The outcome of this process is

- Valid, meaning that any of the (AS1, AS2) or (AS2, AS1) pairs appears in the ASIRIA DB, and AS1 is annotated as a customer of AS2.

- Invalid, if any of the (AS1, AS2) or (AS2, AS1) pairs appears in the ASIRIA DB and the identified c2p relationship goes in the opposite direction, i.e., AS2 has been identified as a customer of AS1.

- Unknown, when no relationship has been inferred between AS1 and AS2.

The resulting value is returned to the route validation, and the rest of the ASPA process continues to determine the status of the path.

The router takes different actions depending if the path is determined as invalid using solely ASPA information or if ASIRIA was also used. In the former case, the route can be discarded (as per ASPA recommendation) while the latter case, the preference of the route can be decreased, so alternative valid routes can be preferred.

## 5.5. Evaluation and experiments

In this section we do a quantitative analysis of the performance of the ASIRIA solution.

### 5.5.1. Performance evaluation of the ASIRIA mechanism

For this purpose, we use two metrics, the *sensitivity*, and the *false positive rate*. The *sensitivity*, the ratio between the leaks detected and the total number of leaks, indicates how many of the actual leaked paths can be detected. The *false positive ratio*, the number of valid routes identified as leaks over the total number of valid routes, reflects the probability of a false positive, i.e., how many paths marked as leaks by our mechanism do not correspond to real leaks. We determine an upper bound for this value. In addition, we compute the sensitivity of the ASIRIA alarm system for detecting leakage events, i.e. an event that generates a set of leaks.

### 5.5.2. Dataset

To compute the metrics in realistic scenarios we use real BGP routing tables from the RIPE Routing Information System (RIS). We retrieved the BGP routing tables from RIS on the 01-03-2021, 08:00. At that point in time, there were 1,601 monitors that periodically dumped BGP information to 20 collectors. Monitors have configured different routing/announcement policies, with some of them announcing their full BGP routing table to the collector. We select the monitors that provide a full IPv4 BGP feed to the collector. Following [74], we identify *full feeds* as those tables containing at least the 75% of the maximum number of prefixes observed in any RIS BGP feed. We identify 302 monitors that comply with the 75% rule, with a mean number of advertised prefixes of 824,000 prefixes and 105,000 different AS path routes. We use these monitors in our analysis.

ASIRIA DB contains 51K P2C relations inferred from the IRR, of which 24K of them appear also in the RIS BGP routing tables analysed. Regarding P2P relationships, there are 46K in ASIRIA DB, with 4.8K appearing in the RIS tables. The total number of different AS pairs appearing in the RIS BGP tables is 373K, so the fraction of this relationships also present in the ASIRIA DB is 7.9%.

### 5.5.3. False positives in ASIRIA leak detection

We analyse the number of false positives in ASIRIA, i.e., the number of leaks reported by ASIRIA that are not real leaks and we compute the false negative rate.

We run the ASIRIA leak detection algorithm in the BGP tables of the monitors in 01-03-2021. According to BGPstream[3], there were no reports for on-going leaks a the time the BGP tables are retrieved (08:00). The leaks detected by ASIRIA are a combination of real

[3]https://bgpstream.com

leaks (even they were not reported) and false negatives. Thus, the computed number is an upper bound of the number of false negatives ASIRIA may generate.

We find that the false positive ratio of the ASIRIA is 0.08%. This is the resulting ration after computing that the mean number of routes marked as a leak per monitor is 0.09% while the fraction of invalid routes (leaks) over the routes that have two or more relationships identified (i.e., the routes for which it is possible to detect an invalid route) is 1.1%. The distribution of these routes over the 302 IPv4 monitors considered is depicted in figure 5.9.

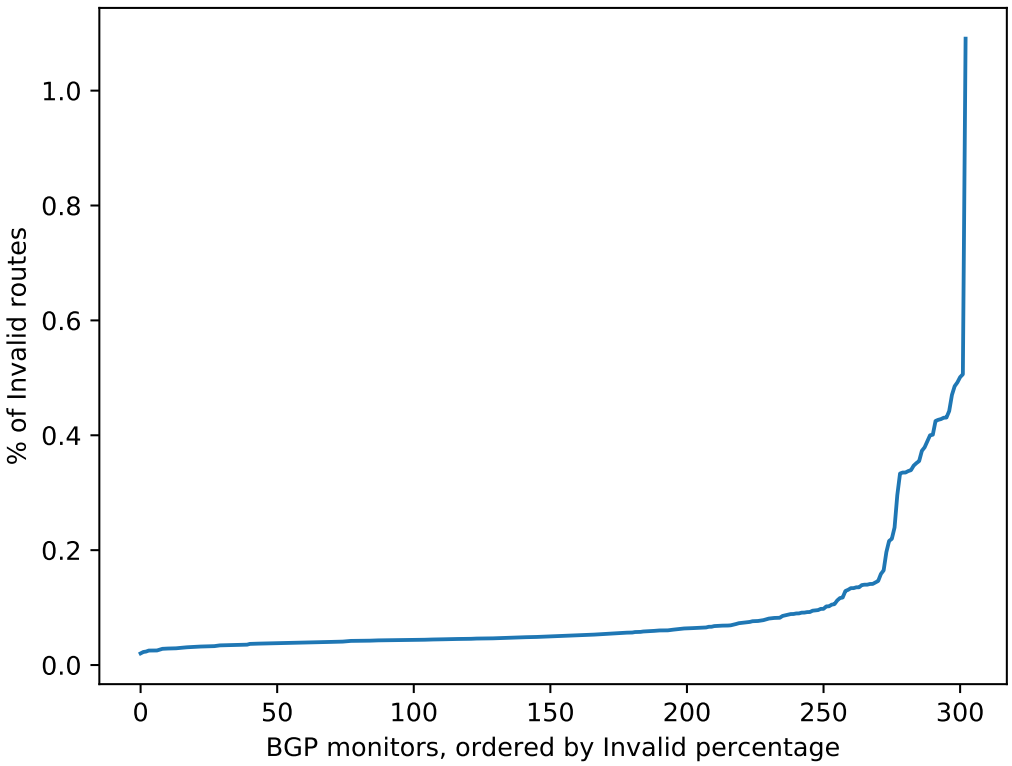


Figure 5.9: Percentage of Invalid routes observed in RIS monitors

We next inspect the routes identified as invalid and we check their variation over time. We search for the same routes in the routing tables of the same RIS monitors, 8 hours before and after. Most of the paths identified as invalid are stable and present in the previous and subsequent routing table snapshots. More precisely, 95% of the routes marked as invalid observed at 08:00 were also present at 00:00, and the coincidence between 08:00 and 16:00 is 98%. We also consider the match with the routing tables captured 10 days after (11/03/2021,

08:00), to observe in this case a match of 85%. This suggest that most of the invalid routes indicated in this set have not changed over time, probably indicating that are legitimate routes (or at least, leaks that did not trigger any corrective action over time.) This is consistent with previous work [120] that has found that 1.5% of the IPv4 routes are valley routes, some of which would be detected as leaks by ASIRIA.

### 5.5.4. ASIRIA leak detection sensitivity

We compute the sensitivity of the proposed solution (i.e., the ratio between the leaks detected and the total number of leaks) when an AS leaks its full routing table to a provider or to a peer. The different leak scenarios considered are depicted in Figure 5.10. In all scenarios depicted, AS1 is running ASIRIA and it is the one detecting the leaks. The scenarios considered include the leakage from a direct customer (AS C2 in the picture), any AS in the customer cone (AS C11 in the picture), a peer (AS P1 in the picture) and any AS in the peer's customer cone (AS CP1 in the picture).

For simplicity of the presentation, we first present the case of a leak from a direct customer as illustrated in Figure 5.11. We generalise for other scenarios later on. In the considered scenario (Figure 5.11), a router (`L`) in a customer AS (the *leaky neighbour*) leaks its whole routing table to an ASIRIA-enabled router (`A`) in its provider AS. We compute the level of protection provided by ASIRIA for this scenario. We use the BGP tables retrieved from RIS as the BGP tables from the *leaky neighbour* AS that are leaked by router A to L. When `A` advertises all its routes to `L`, only those corresponding to peers or providers of `L` are actual leaks. `L`'s own routes and the routes from its customers are legitimate announcements to `A`. We identify all these routes using the relationships inferred by CAIDA to determine which neighbors of `L` that are its customers and we remove them from the leak count. The remaining routes are considered to be leaks. We next compute the sensitivity of ASIRIA when detecting leaks in router `A`.

In average across all the 302 experiments (each one corresponding to the leakage of the routing table of a different RIS monitor), the ASIRIA-enabled router is capable of detecting 17.7% of the routes leaked by the neighbour. The distribution for different experiments is shown in figure 5.12. We observe that there are important differences between experiments regarding the leak detection capability. In six cases, ASIRIA is able to detect more than 90% of the routes as leaks, and in 20% of the cases, ASIRIA is able to detect at least 40% of the leaks. In all these cases, ASIRIA provides immediate significant gains. On the other extreme, some cases fail to detect a significant amount of routes.

We mentioned that most of the information comes from the RIPE IRR and because of this, we expect that most of the relationship inferred involve European networks. Indeed,

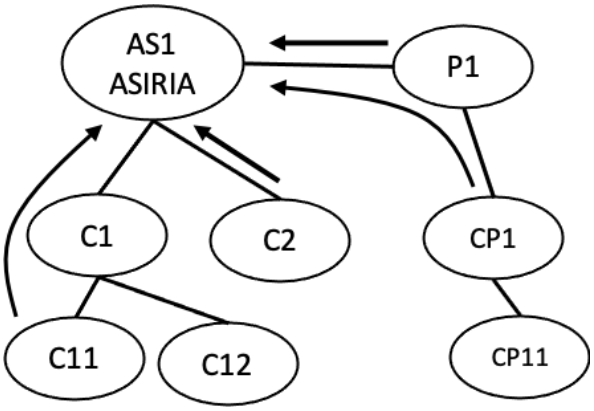


Figure 5.10: Leak scenarios

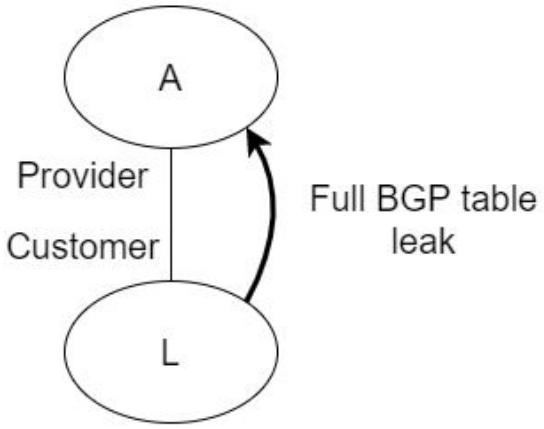


Figure 5.11: Router leaking its routes to a provider

experiments using routing tables from monitors located in regions outside RIPE exhibit a much lower leak capability detection. The ratio of leaks detected for the other regions are: USA: 2.7%, ASIA: 5.85%, LACN: 9.16% and AFR: 7.39%.

The analysis presented shows the case in which a direct neighbor of an ASIRIA router originates the leak. However, these results apply to any leak coming through a customer AS or a peering AS. This means that ASIRIA detects leaks generated by an AS within the customer's cone of the detecting AS or within the customer cone of the peer of the detecting AS, even though none of the intermediate routers are applying ASIRIA. The reasoning is straightforward: a route received from the customer cone or the customer cone of a peer is identified as upstream, so any P2C/P2P relationship breaks the validity rules for such route. Thus, the previous figures also represent the capacity of an ASIRIA router to detect leaked paths if the leaker is located in its customer cone or in the peer's customer cone.

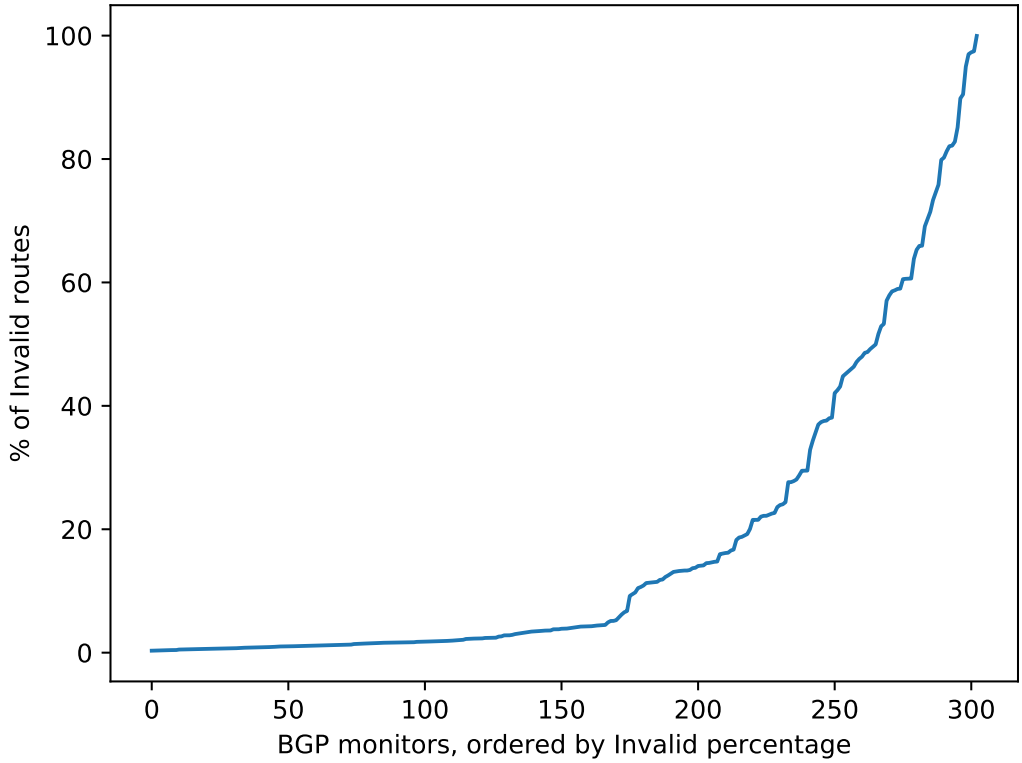


Figure 5.12: Percentage of leaks detected in different experiments of the client AS leaking its full routing table to its provider. Different experiments account for different routing tables used fro the leaky neighbour.

### 5.5.5. Sensitivity of the ASIRIA leakage event alarm system

In addition to detecting and marking individual routes leaks, ASIRIA is also able to detect leakage events and trigger alarms that can alert the network administrator that a leakage event is ongoing. An ASIRIA alarm is triggered when the number of leaks present in the BGP routing table of the ASIRIA enabled router increments beyond a configured threshold **Th** within a short period of time **p** (i.e., at least **Th**% increase **p** minutes or less).

In order to set a value for **Th**, we compute the number of invalid paths present in the analyzed BGP routing tables in a moment when there was no reports of any leakage event (steady-state) and compare it the the number of paths identified as invalid when a full BGP is leaked from the neighbour in the scenario depicted in Figure 5.11. IWe find that the mean number of invalid paths per BGP routing table in steady state is 96 (using the data from

section 5.5.3), while the mean number of invalid paths detected after a leakage of a full routing table is 18,454 (using the data from Section 5.5.4). Setting **Th** as closer as possible to the value of invalid paths present in steady state would allow ASIRIA to detect leakage events that involve few routes, but it may also result in a high number of false alarms. However, given the very large gap between the number of invalid paths in steady state and the number of invalid paths in the case of a full BGP feed, we set **Th** to a 100% increase in order to be conservative.

We next compute the sensitivity of the ASIRIA alarm system for a number of offending ASes and a number of victims. As before, we consider the scenario depicted in Figure 5.11 and we compute the number of false positives (using the methodology described in section 5.5.3) and detected leaks (using the methodology described in Section 5.5.4) for the different combinations of offending AS and ISP (i.e., we use all possible combinations of two routing tables from the selected RIS monitors and we use one as the offending AS leaking its full routing table and the other as being the victim provider AS receiving the leaked routes). For each combination, we compute if the ASIRIA alarm using the configured threshold (100% increase in the number of leaks) would indeed detect the leakage event.

Figure 5.13 plots the distribution of the number of leakage events detected by the ISPs running ASIRIA. We can see that 74% of the ISPs detect 100% of the leakage events, that 90% of the ISPs detect at least 99% of the leakage events and that 99.9% of the ISPs detect at least 83% of the leakage events.

We next look into the temporal dimension of the leakage event and set the duration of the time bin **p** used to detect them. According to [121], it takes up to 60 seconds to receive and process a full routing table from a BGP peer in normal conditions. When the leak is originated farther away, the announcements of the different paths will be more separated in time from each other. In order to account for that, and enable ASIRIA to detect remote leakage event, we set **p** to 5 minutes.

We verify if the proposed time interval **p** triggers false alarms when used with the proposed threshold **Th**. We analyze the BGP feed of the 302 monitors in the interval from 10:00 AM until 16:00 AM on the 01-03-2021. No leakage was reported during with period. Then, for every 5 minute bin of the analyzed period, we compute the increase in the number of invalid path as a ratio of the invalid paths present in steady state in the BGP table of each monitor. The maximum increase observed across all the monitors during this period was of 0.2%, far from the 100% defined for the threshold **Th**.

With all this, we set **Th** to 100% and **p** to 5 minutes, the ASIRIA leakage event alarm system allows 90% of the monitors to detect at least 99% of leakage events generated by a neighbour.

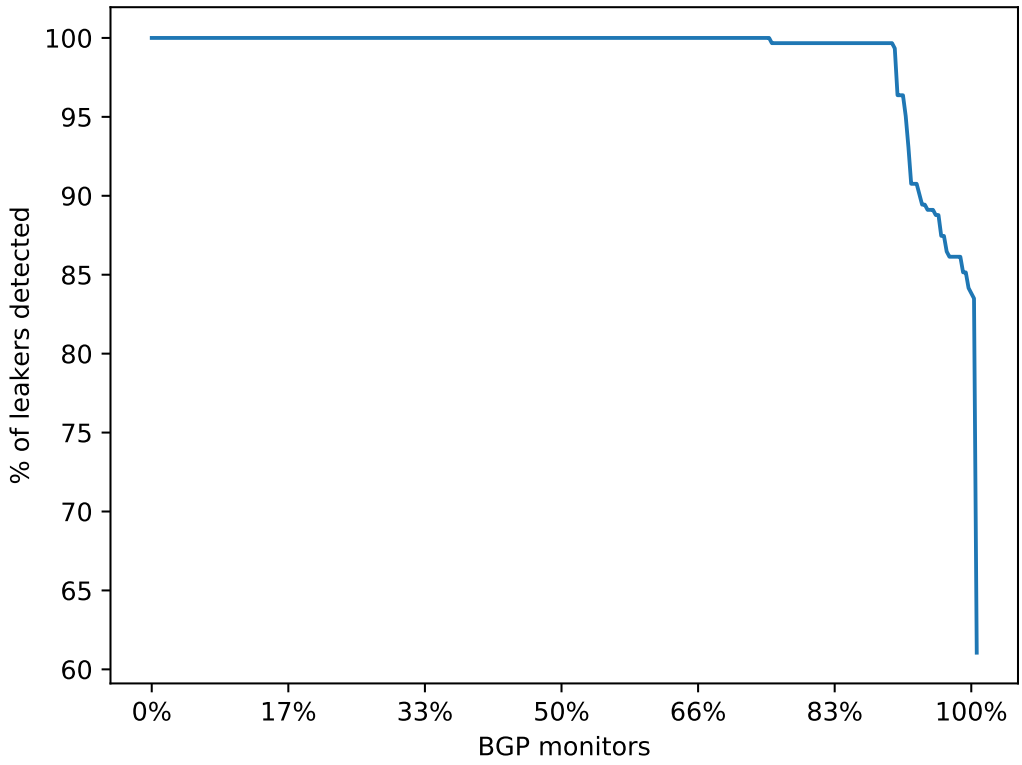


Figure 5.13: Distribution of the number of leakage events detected using ASIRIA across the different experiments performed.

### 5.5.6. Route leak detection systems key characteristics

We conclude the chapter with an overview of the key characteristic that a route leak detection mechanisms should have. As part of a survey among 75 network operators on BGP prefix hijacking done in 2018 [122], ISPs were polled regarding the importance of different characteristics a system to provide protection against Border Gateway Protocol Security (BGPsec) incidents should have. We next enumerate the top characteristics mentioned by the operators, from most valued to less valued and we expand on how ASIRIA performs from the perspective of each of top four features appreciated by the ISPs in the survey.

1. **Effectiveness of mitigation**. Unsurprisingly, the most important feature for ISPs is the effectiveness of the mechanism. The problem is that due to the distributed nature of the problem, the effectiveness of any solution depends on its degree of adoption across the Internet, usually providing little or no value for early adopters. ASIRIA

relies on existing information in the IRR to infer relationships. Because of this, it can protect early adopters. As presented, ASIRIA allows 90% of the analyzed ASes to detect more than 99% of the leakage events. In addition, ASIRIA allows early adopters to detect 17% of leaked routes in average. The effectiveness of ASIRIA will increase as more ASes populate the IRR, which is actively being promoted by initiatives such as Mutually Agreed Norms for Routing Security (MANRS). In the survey, 80% of the queried ISPs were willing to disclose they routing policies to increase BGPsec.

2. **Fast mitigation**. A router running ASIRIA has the information used to validate the paths locally available, so invalid paths are detected upon its reception as part of the BGP route selection process. In the case of events, we have set the route leakage event detection to use a 5 min period, but if the number of invalid paths reached the configured threshold **Th** earlier, the detection occurs faster. This is likely to be the case of major route leak events, when many paths are affected.

3. **Self-managed/operated**. 61% of the ISPs in the survey declared to rely on notifications from third-party services to detect BGP route hijacking incidents, and another 61% of ISPs declared that they would rather avoid outsourcing these kind of functions. A distinctive characteristic of ASIRIA is that it runs locally and it is not a service provided by a third party. It only relies on public IRR information that is periodically updated, but no real time access to the IRR is required. ASIRIA is able to provide effectiveness to early providers, a feature typically only available to solutions provided by third parties, while being locally operated.

4. **Ease of operating/troubleshooting**. As 61% of ISPs in the survey already use third party notification of relevant events, the ASIRIA route leak alarm may fit in their existing operational workflow, although in this case the service could be operated in-house. ASIRIA detects both leakage events and specific invalid routes. By marking specific routes as invalid, ASIRIA not only avoids those routes, but also provides information to the network administrator to troubleshoot and determine the offending AS causing the route leak.

Also, according to the same survey 80operators that participated in the poll stated that they would be willing to share information about their routing policies. This data point sheds a positive light about the potential protection that can be achieved by ASIRIA in the future.

# 6 Conclusions and future work

Blockchain and smart contract, with their decentralized trust and security properties, provide the development of automatic system that operate with no need of intermediaries, at low cost, with fast update times and high ledger visibility without sacrificing security. Blockchain technology represents a never had opportunity to improve the management and exchange of sensible information and resources. In this research work we investigated the application of this technology for the management of internet resources. Additionally we have proposed a novel mechanism to perform route leak detection.

In section 4.2 we have presented the design of InBlock, a decentralized autonomous organization that is capable of performing respectively IPv6/IPv4 global registry functions without human intervention. We have designed the mechanism to create the incentives so that the system can autonomously prevent different form of abuses, including stock-pilling and other wasteful techniques, and thus, preserve the address space. Because the InBlock is decentralized, as its execution is performed by the different miners all around the planet, the InBlock also addresses the identified jurisdictional overflow problem. The clear and transparent rules defined may appeal to organizations facing legal uncertainty costs due to their exposition to multiple legal jurisdictions. Moreover, InBlock's prefix allocation incurs in very low operational cost, tens of US$ (that is increased to serve address conservation purposes) and very low delay, tens of minutes, as discussed in section 2.1.2. There are many different ways that blockchains can be used to build a global IP address registry. In particular, InBlock implements a registry that is not under control of any single entity, implying that no single entity can prevent another entity to obtain IP address allocations. InBlock provides quite strong privacy guarantees and censorship resistance. Furthermore InBlock implements an alternative trust model to the hierarchical one currently provided by the RPKI. Nevertheless, as a trade-off, InBlock gives up the traceability and enforcement features. This means that the IP address registry functions provided by InBlock cannot be

used to trace the identity of the holders of the resources if they do not voluntarily include contact information in the registry. Similarly, the InBlock cannot be used to restrict the Internet access to any entity by revoking their address allocations (actually it is a goal of the InBlock to prevent this situation). The InBlock design described here aims to achieve some defined goals (uniqueness, stockpiling prevention, aggregation preservation, etc.) by means of some mechanism (blockchain consensus, pricing, limits on the number of AS number involved in the delegations, etc.) The extent to which these goals can be achieved with these measures can only be derived from real experience.

Within InBlock in sections 4.3 and 4.4 we have presented the implementation and the evaluation of its two Proof of Concept (PoC), InBlock6 and InBlock4. While InBlock6 is the implementation of the InBlock design, InBlock4 inherits the same features of InBlock6 but for the IPv4 address space and in addition it provides a blockchain-based alternative to RPKI for the provision of Route Origin Validation (ROV) for BGP, offering a mechanism to register living resources (e.g., already assigned resources) into the blockchain. Additionally InBlock4 is compatible with BGPsec, it can be bootstrapped using the information in the RPKI and it can coexists with RPKI-based route origin validation. Hence we have described both InBlock6 and InBlock4 implementation and experimental evaluation, along with contract deployment, prefix management (allocation, delegation, recovery and route information update), access by third parties and key rollover functions. Both the InBlock PoC exhibits very low operational costs, in the order of tens of US$, several orders of magnitude than the value of the assets it manages. This cost is unbeatable for the human-based organizations that currently perform address allocation functions. The time required to perform an address allocation is in the order of minutes, much better also than the time achievable with current practices. Besides, third parties have immediate access to registry information, improving the overall security of the allocation system.

In section 5.2 we have presented the Internet Routing Blockchain (IRB), an Hyperledger Fabric (HF) consortium blockchain based solution to preserve the distributed nature of IRRs which provides information consistency and stall-ness prevention by design. Has been shown that the data stored in IRB can be used by novel route leaks prevention solution such as ASPA and ASIRIA. The experimental evaluation shows that the prototype scales linearly with respect to the amount of information to be processed.

In section 5.4 we have presented ASIRIA, a novel mechanism to provide protection against route leaks based on *drop-lists*. ASIRIA is capable of inferring the type of relationships between ASes from the routing policies declared and available in the IRR. ASIRIA identifies and avoids leaks handling this information about ASes' relationships. We have showed that is possible to detect leaked routes by analyzing the relationships between consecutive ASes

present in the AS_PATH attribute of a BGP route with an high level of confidence. ASIRIA identifies routes containing non *valley-free* paths and adds them in the *drop-list*, in order to avoid their usage if alternative routes are available. Additionally ASIRIA keeps track of the number of leaks detected and employs this information to build a leakage event alarm mechanism that allows ASIRIA-enabled routers to detect leakage events and notify the network administrator of suspicious behaviour. Finally we have showed the quantitative analysis of the performance of the ASIRIA solution.

In summary in this thesis we have explored the potential of the application of blockchain technology for the management of internet resources and we have proposed a novel mechanism that provides route leak detection. We note that there is not enough experience in the use of blockchains for solving real-world problems. Thereby we believe the research conducted in this thesis could provide useful hand-on experience about blockchain-based organizations for the internet resource management and can be a starting point for future research on the topic. We conclude this thesis with an overview on the possible future evolution of the presented research work and, finally, we discuss an experiment proposal where InBlock is employed to allocate a small set of IP addresses.

## 6.1. Future work

As an evolution of the Namecoin [29] research work, discussed in chapter 3, other blockchain based Domain Name System (DNS) have been proposed like Unstoppable Domains [123]. Unstoppable Domains propose new domain names extensions (e.g., .zil, .crypto, .coin, .wallet, .bitcoin, .x ,.888, .nft, .dao, and .blockchain*) and provides a Non-Fungible token (NFT) domain market and several browser extensions to guarantee the access to new proposed domains. NFTs are tokens that can be used to represent ownership of unique items. They can only have one official owner at a time and they're secured by the Ethereum blockchain. With a sum of more than 1.4 milions of registered domain, Unstoppable Domains is proving the NFT technology value outside of its main current employment field, that is the artistic sector. Since NFT can be used to represent ownership of any unique asset would be natural to consider, as an evolution of InBlock, an IP address registry that manages address prefixes in the form of NFT. The employment of loanable NFT can solve several problems specifically related to the conservation and preservation of the IP address space while providing a simplified management of the resources.

Furthermore in the IRB research work, all the experiments are run in a local environment composed of docker containers with all users identities not actually linked to real Autonomous System Number (ASN) holders identities. A possible improvement can be a physical

distributed deployment of the blockchain architecture to measure and compare the times that have been computed just locally with an addition of a special management of users identity employing a Lightweight Directory Access Protocol (LDAP) server that would provide the possibility of transferring the holdership rights, acquired in the current used system, to the IRB. Besides, with an IRB evolution, can be worth to compute the time required by ASIRIA to retrieve and process the information stored in the blockchain system, in order to compare it to the current solution that uses the IRR information.

Finally we present an experiment proposal where InBlock is used to allocate a small IP address block out of the global address space for an experiment and to create one or more blockchain-based organizations to manage the allocations out of that address block, with a predefined lifetime. We next describe the objectives of the experiment, and how to run it in controlled, yet realistic, conditions.

### 6.1.1. Proposal for a global Inblock experimental deployment

The deployment model we envision for InBlock is as follows. To operate, InBlock requires a block of globally unique IPv6 or IPv4 addresses to allocate. Once the block is assigned to the InBlock, it may prove to be hard to recover, so it is probably wise to be conservative and assign a small block (e.g., a /20 if IPv6). InBlock can then execute and perform allocations out of this initial block. If InBlock works as expected and all the addresses out of this initial block are allocated, it would be possible to launch other InBlock instances, each of them managing a separated address block. This deployment model allows to start with a small portion of the address space managed by the InBlock and grow if the demand increases, without jeopardizing the fate of a large address block from the start.

Different InBlock instances do not necessarily have to be equal, neither in size nor in encoded properties. Experience managing previous InBlock instances can inform modifications to be implemented in future ones. Moreover, it is also possible to envision different entities launching their own InBlock instances, e.g., if this service is proven to be valuable, different Regional Internet Registries (RIRs) may decide to run their own InBlock instance. The deployment model of InBlock naturally supports this.

As the blockchain technology is fairly new, constantly evolving and probably susceptible to bugs, and the IP address allocation apparatus is a cornerstone of the Internet, we are not proposing to create an operational IPv6/IPv4 registry using InBlock. Instead, we are promoting an experiment with a limited scope, to gain a better understanding of how blockchain technology can be used to provide registry services for Internet resources and moreover, if the notion of a distributed autonomous registry is feasible.

Through the proposed experiment, we believe we can gain a better understanding of the

following aspects:

- Whether the proposed approached based on both a yearly fee and the abundance argument is enough to deter stockpiling.
- Whether the addresses obtained from the InBlock are actually used and announced in the Internet.
- Whether the blocks are announced as they are assigned, or extra deaggregation is observed.
- Whether a secondary market of addresses emerges.
- Whether the contact information of the allocated blocks is completed and updated with sound contents.
- How changes in Ethereum affect the InBlock service. In particular, when bugs appear, how much they affect the InBlock service and whether it is possible to cope with them. Also, how technological leaps in Ethereum affect the InBlock service (e.g., if Ethereum moves from Proof of Work (PoW) to Proof of Stake (PoS), how this affects InBlock).

The experiment, by definition, has a limited lifetime. The lifetime should be defined in advance, before starting the experiment. A reasonable time frame could be five years. If the experiment is a success, the lifetime of the experiment can be extended and the allocations made renewable for as long as the holders want to renew them. If the experiment is terminated, the allocations made through the InBlock should be transferred to one of the existent Internet registries upon its termination. This would allow organizations participating in the InBlock experiment to avoid the cost of renumbering regardless the result of the experiment. With this conditions, the InBlock experiment is likely to be more realistic, as more organizations may use the resources obtained through the InBlock in real operations.

The experiment should include some safeguards in case an unexpected behavior is observed. For instance, it would be beneficial to rate limit the number of allocations made. If the rate in the applications is excessive, the experiments should be paused and the control passed to human supervisors.

In order to run the InBlock experiment, a pool of globally unique addresses is needed. The organizations that can provide the required pool of addresses and run the experiment are:

- The Internet Engineering Task Force (IETF) (80% of the IPv6 address space is "Reserved for the IETF" [124], and it is up to the IETF to instruct Internet Assigned Number Authority (IANA) to allocate an address block for the InBlock experiment),
- Internet Corporation for Assigned Names and Numbers (ICANN), through the IANA,
- The RIRs, either one of them on its own, or jointly through the Number Resource Organization (NRO)
- A National Internet Registry (NIR).

In addition of providing the addresses required to perform the experiment, the organization running the experiment should also be capable of properly designing and implementing it.

In this work we proposed InBlock that could serve as a starting point for that debate, but in order for the experiment to be useful, the Internet community should be actively engaged in the ultimate design of the experiment. Similarly, once the experiment has been defined, it needs to be implemented and tested. The resulting implementation should be reviewed by the community to make sure that there are no errors or backdoors. We believe that there are several organizations who have the capabilities to run the proposed experiment, such as the IETF or the RIRs. Moreover, if the experiment is proven to be successful, the IETF and/or the RIRs could naturally have a stake in the distributed autonomous organization.

# References


[1] S. Angieri, A. García-Martínez, B. Liu, Z. Yan, C. Wang, and M. Bagnulo, "A distributed autonomous organization for internet address management," *IEEE Transactions on Engineering Management*, vol. 67, no. 4, pp. 1459–1475, 2020.

[2] A. García-Martínez, S. Angieri, B. Liu, F. Yang, and M. Bagnulo, "Design and implementation of inblock—a distributed ip address registration system," *IEEE Systems Journal*, vol. 15, no. 3, pp. 3528–3539, 2021.

[3] S. Angieri, M. Bagnulo, A. García-Martínez, B. Liu, and X. Wei, "Inblock4: Blockchain-based route origin validation," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2020, pp. 291–296.

[4] S. Angieri, M. Bagnulo, and A. García-Martínez, "Internet routing blockchain: an hyperledger fabric consortium blockchain for internet routing registries," *This work has been submitted for publication.*

[5] S. Angieri, M. Bagnulo, A. García-Martínez, A. Lutu, and J. Yang, "Practicable route leak detection and prevention with asiria," *This work has been submitted for publication.*

[6] Merit, "Overview of the irr," http://www.irr.net/docs/overview.html, [Online; accessed 25-Jun-2021].

[7] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2009. [Online]. Available: http://www.bitcoin.org/bitcoin.pdf

[8] S. DAVIDSON, P. DE FILIPPI, and J. POTTS, "Blockchains and the economic institutions of capitalism," *Journal of Institutional Economics*, vol. 14, no. 4, p. 639–658, 2018.

[9] C. Team, "CoinMarketCap," https://coinmarketcap.com/all/views/all/, May 2019.

[10] J. Desjardins, "All of the World's Money and Markets in One Visualization," http://money.visualcapitalist.com/worlds-money-markets-one-visualization-2017/, October 2017.

[11] K. Fanning and D. P. Centers, "Blockchain and its coming impact on financial services," *Journal of Corporate Accounting & Finance*, vol. 27, no. 5, pp. 53–57, 2016.

[12] A. Savelyev, "Copyright in the blockchain era: Promises and challenges," *Computer Law & Security Review*, vol. 34, 12 2017.

[13] P. De Filippi and S. Hassan, "Blockchain technology as a regulatory technology: From code is law to law is code," *First Monday*, vol. 21, 11 2016.

[14] H. Zhao, "ITU and Internet Governance," *ITU document WG-WSIS-7/6 Rev 1*, 2004.

[15] L. R. E. Heilman, D. Cooper and S. Goldberg, "From the Consent of the Routed: Improving the Transparency of the RPKI," *Proceedings of the 2014 ACM conference on SIGCOMM*, Aug 2014.

[16] K. Sriram, D. Montgomery, D. R. McPherson, E. Osterweil, and B. Dickson, "Problem Definition and Classification of BGP Route Leaks," RFC 7908, Jun. 2016. [Online]. Available: https://rfc-editor.org/rfc/rfc7908.txt

[17] RIPE, "IPv6 Address Allocation and Assignment Policy," https://www.ripe.net/publications/docs/ripe-699#minimum_allocation, May 2018.

[18] V. Buterin, "Ethereum White Paper," https://github.com/ethereum/wiki/wiki/White-Paper, May 2018.

[19] Z. Zheng, S. Xie, H. Dai, X. Chen and H. Wang, "An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends. IEEE International," *BigData Congress*, 2017.

[20] D. Mingxiao, M. Xiaofeng, Z. Zhe, W. Xiangwei and C. Qijun, "A review on consensus algorithm of blockchain," *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2017.

[21] M. E. Peck, "How Smart Contracts Work," *IEEE, Spectrum*, 2017-09.

[22] L. Yu, G. Li, Y. Yao, C. Hu and W. Tsai and E. Deng, "Smart-Contract Execution with Concurrent Block Building," *IEEE Symposium on Service-Oriented System Engineering*, 2017.

[23] R. Bhadoria, A. Nimbalkar, and N. Saxena, *On the Role of Blockchain Technology in the Internet of Things*. Springer, 2020, pp. 129–140.

[24] Ethereum Community, "Proof of Staket," 2021, accessed: 2021-11-14. [Online]. Available: https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/

[25] V. Buterin, "DAOs, DACs, DAs and More: An Incomplete Terminology Guide," https://blog.ethereum.org/2014/05/06/daos-dacs-das-and-more-an-incomplete-terminology-guide/, May 2018.

[26] R. Bhadoria, Y. Arora, and K. Gautam, *Blockchain Hands on for Developing Genesis Block*. Springer, 2020, pp. 269–278.

[27] A. Hari and T.V. Lakshman, "The Internet Blockchain: A Distributed, Tamper-Resistant Transaction Framework for the Internet," *HotNets-XV*, 2016.

[28] J. Palisse, A. Rodriguez-Natal, V. Ermagan and A. Cabellos, "An analysis of the applicability of blockchain to secure IP addresses allocation, delegation and bindings," *IETF draft-paillisse-sidrops-blockchain-01, work in progress*, 2017-10.

[29] H. Kalodner, M. Carlsten, P. Ellenbogen, J. Bonneau and A. Narayanan, "An empirical study of Namecoin and lessons for decentralized namespace design," *Workshop on the Economics of Information Security (WEIS)*, Jun 2015.

[30] V. Buterin and V. Griffith, "Casper the Friendly Finality Gadget," https://vitalik.ca/files/casper_note.html, May 2018.

[31] M. Suiche, "The $280M Ethereum's Parity bug," https://blog.comae.io/the-280m-ethereums-bug-f28e5de43513, May 2018.

[32] "Proof-Of-Stake," https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQ, May 2018.

[33] "ETH Gas Station," https://ethgasstation.info/, May 2018.

[34] V. Buterin, "On slow and Fast Block Times," https://blog.ethereum.org/2015/09/14/on-slow-and-fast-block-times/, May 2018.

[35] "Gas Price Oracle," https://github.com/ethereum/go-ethereum/wiki/Gas-Price-Oracle, May 2018.

[36] IBM, "What is hyperledger fabric?" https://www.ibm.com/topics/hyperledger, [Online; accessed 6-Jul-2021].

[37] H. Community, "hyperledger-fabric-ca documentation," https://readthedocs.org/projects/hyperledger-fabric-ca/downloads/pdf/latest/, [Online; accessed 31-May-2021].

[38] ——, "Peers," https://hyperledger-fabric.readthedocs.io/en/release-2.2/peers/peers.html, [Online; accessed 31-May-2021].

[39] ——, "Couchdb as the state database," https://hyperledger-fabric.readthedocs.io/en/release-2.2/couchdb_as_state_database.html, [Online; accessed 31-May-2021].

[40] ——, "Transaction flow," https://hyperledger-fabric.readthedocs.io/en/release-2.2/txflow.html, [Online; accessed 31-May-2021].

[41] M. Hogewoning, "A review of blockchain applicability to internet number resources," https://labs.ripe.net/author/marco_hogewoning/a-review-of-blockchain-applicability-to-internet-number-resources/, [Online; accessed 6-Jul-2021].

[42] H. Community, "Private data," https://hyperledger-fabric.readthedocs.io/en/release-2.2/private-data/private-data.html, [Online; accessed 9-Jul-2021].

[43] R. Housley, J. Curran, G. Huston and D. Conrad, "The Internet Numbers Registry System," IETF RFC 7020, Aug. 2013.

[44] M. Mueller, "Critical resource: An institutional economics of the Internet addressing-routing space," *Telecommunications Policy*, 2010.

[45] ICANN, "Stewardship of IANA Functions Transitions to Global Internet Community as Contract with U.S. Government Ends," https://www.icann.org/news/announcement-2016-10-01-en, Jul 2018.

[46] ——, "Stewardship of IANA Functions Transitions to Global Internet Community as Contract with U.S. Government Ends," https://www.icann.org/resources/pages/governance/litigation-en, May 2018.

[47] APNIC, "IPv6 Address Allocation and Assignment Policy," https://www.apnic.net/community/policy/ipv6-address-policy_obsolete/, May 2018.

[48] ARIN, "IPv6 Address Allocation and Assignment Policy," https://www.arin.net/vault/policy/archive/ipv6_policy.html, May 2018.

[49] LACNIC, "IPv6 Address Allocation and Assignment Policy," http://www.lacnic.net/684/2/lacnic/4-ipv6-address-allocation-and-assignment-policies, May 2018.

[50] AFRINIC, "IPv6 Address Allocation and Assignment Policy," https://afrinic.net/fr/library/policies/122-afpub-2004-v6-001, May 2018.

[51] RIPE, "What are Provider Aggregatable (PA) addresses and Provider Independent (PI) addresses?" https://www.ripe.net/participate/member-support/copy_of_faqs/isp-related-questions/pa-pi, May 2018.

[52] B. Kuerbis and M. Mueller, "Internet routing registries, data governance and security," *Journal of Cyber Policy*, 2017.

[53] B. Carpenter and S. Jiang, "Significance of IPv6 Interface Identifiers," IETF RFC 7136, Feb. 2014.

[54] M. Lepinski and S. Kent, "An Infrastructure to Support Secure Internet Routing," IETF RFC 1884, Feb 1995.

[55] NIST, "RPKI Monitor NIST," https://rpki-monitor.antd.nist.gov/, Dec 2018.

[56] Number Resource Organization, "Regional Internet Registries are preparing to deploy "All Resources" RPKI Service," https://www.nro.net/regional-internet-registries-are-preparing-to-deploy-all-resources-rpki-service/, May 2018.

[57] P. Sermpezis, V. Kotronis, P. Gigis, X. Dimitropoulos, D. Cicalese, A. King, and A. Dainotti, "ARTEMIS: Neutralizing BGP Hijacking within a Minute," *IEEE/ACM Transactions on Networking*, Oct 2018.

[58] A. Toonk, "Today's BGP leak in Brazil," https://bgpmon.net/todays-bgp-leak-in-brazil/, Oct 2017, accessed: 2020-02-18.

[59] S. Kent and D. Ma, "Adverse Actions by a Certification Authority (CA) or Repository Manager in the Resource Public Key Infrastructure (RPKI)," RFC 8211, Sep. 2017. [Online]. Available: https://rfc-editor.org/rfc/rfc8211.txt

[60] Q. Xing, B. Wang, and X. Wang, "BGPcoin: Blockchain-based internet number resource authority and BGP security solution," *Symmetry*, vol. 10, no. 9, p. 408, 2018.

[61] I. Sfirakis and V. Kotronis, "Validating IP prefixes and AS-paths with blockchains," *arXiv preprint arXiv:1906.03172*, 2019.

[62] P. McCorry, S. F. Shahandashti, and F. Hao, "A smart contract for boardroom voting with maximum voter privacy," in *Financial Cryptography and Data Security*. Springer, 2017, pp. 357–375.

[63] M. Al-Bassam, "SCPKI: A Smart Contract-based PKI and Identity System," in *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts*, 2017, pp. 35–40.

[64] R. Hinden and S. Deering, "IP Version 6 Addressing Architecture," IETF RFC 1884, 1995.

[65] M. Mueller and Y. Kim, "Economic Factors in the allocation of IP addresses," *International Telecommunication Union*, 2009.

[66] MANRS, "MANRS Implementation Guide, Filtering," https://www.manrs.org/isps/guide/filtering/, accessed: 2020-11-11.

[67] A. Azimov, E. Bogomazov, R. Bush, K. Patel, and K. Sriram, "Route Leak Prevention using Roles in Update and Open messages," Internet Engineering Task Force, Internet-Draft draft-ietf-idr-bgp-open-policy-15, Jan. 2021, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-idr-bgp-open-policy-15

[68] A. Cohen, Y. Gilad, A. Herzberg, and M. Schapira, "Jumpstarting bgp security with path-end validation," in *Proceedings of the 2016 ACM SIGCOMM Conference*, ser. SIGCOMM '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 342 355. [Online]. Available: https://doi.org/10.1145/2934872.2934883

[69] B. Kuerbis and M. Mueller, "Internet routing registries, data governance, and security," *Journal of Cyber Policy*, vol. 2, no. 1, pp. 64–81, 2017. [Online]. Available: https://doi.org/10.1080/23738871.2017.1295092

[70] D. Chen, Y. Ba, H. Qiu, J. Zhu, and Q. Wang, "Isrchain: Achieving efficient interdomain secure routing with blockchain," *Computers & Electrical Engineering*, vol. 83, p. 106584, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045790619316350

[71] J. Yue, Y. Qin, S. Gao, W. Su, G. He, and N. Liu, "A privacy-preserving route leak protection mechanism based on blockchain," in *2021 IEEE International Conference on Information Communication and Software Engineering (ICICSE)*, 2021, pp. 264–269.

[72] M. F. Galmés, R. C. Aumatell, A. Cabellos-Aparicio, S. Ren, X. Wei, and B. Liu, "Preventing route leaks using a decentralized approach," in *2020 IFIP Networking Conference (Networking)*, 2020, pp. 509–513.

[73] A. Azimov, E. Bogomazov, R. Bush, K. Patel, and J. Snijders, "Verification of AS_PATH Using the Resource Certificate Public Key Infrastructure and Autonomous System Provider Authorization," Internet Engineering Task Force, Internet-Draft draft-ietf-sidrops-aspa-verification-07, Feb. 2021, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-sidrops-aspa-verification-07

[74] C. Testart, P. Richter, A. King, A. Dainotti, and D. Clark, "To filter or not to filter: Measuring the benefits of registering in the rpki today," in *International Conference on Passive and Active Network Measurement*. Springer, 2020, pp. 71–87.

[75] M. Lepinski and S. Kent, "An Infrastructure to Support Secure Internet Routing," RFC 6480, Feb. 2012. [Online]. Available: https://rfc-editor.org/rfc/rfc6480.txt

[76] V. Buterin, "The Meaning of Decentralization," https://medium.com/@VitalikButerin/the-meaning-of-decentralization-a0c92b76a274, Feb 2017.

[77] M. Lepinski and K. Sriram, "BGPsec Protocol Specification," RFC 8205, Sep. 2017. [Online]. Available: https://rfc-editor.org/rfc/rfc8205.txt

[78] Cisco, "All Events for BGP Stream," https://bgpstream.com/, accessed: 2020-02-18.

[79] Y. Gilad, A. Cohen, A. Herzberg, M. Schapira, and H. Shulman, "Are We There Yet? On RPKI's Deployment and Security," in *NDSS 2017*. The Internet Society, 2017. [Online]. Available: https://www.ndss-symposium.org/ndss2017/ndss-2017-programme/are-we-there-yet-rpkis-deployment-and-security/

[80] B. Kuerbis and M. Mueller, "Negotiating a new governance hierarchy: An analysis of the conflicting incentives to secure internet routing," *Communications and Strategies*, vol. 81, 03 2011.

[81] D. Cooper, E. Heilman, K. Brogle, L. Reyzin, and S. Goldberg, "On the risk of misbehaving RPKI authorities," in *HotNets-XII*, 2013. [Online]. Available: https://doi.org/10.1145/2535771.2535787

[82] J. Arkko, B. Trammell, M. Nottingham, C. Huitema, M. Thomson, J. Tantsura, and N. ten Oever, "Considerations on Internet Consolidation and the Internet Architecture," Working Draft, IETF, Internet-Draft draft-arkko-iab-internet-consolidation-02, July 2019.

[83] I. Society, "Consolidation in the Internet Economy," ISOC, ISOC Global Internet Report, 2019. [Online]. Available: https://future.internetsociety.org/2019/wp-content/uploads/sites/2/2019/04/InternetSociety-GlobalInternetReport-ConsolidationintheInternetEconomy.pdf

[84] J. Arkko, "Centralised Architectures in Internet Infrastructure," Working Draft, IETF Secretariat, Internet-Draft draft-arkko-arch-infrastructure-centralisation-00, November 2019.

[85] K. Hubbard, M. Kosters, D. Conrad, D. Karrenberg and J. Postel, "Internet Registry IP Allocation Guidelines," IETF RFC 2050, 1996.

[86] A. Durand and C. Huitema, "The Host-Density Ratio for Address Assignment Efficiency: An update on the H ratio," IETF RFC 3194, 2001.

[87] WorldBank, "GDP, current US$ ," https://data.worldbank.org/indicator/NY.GDP.MKTP.CD, Dec 2018.

[88] L. Levin and Stephen Schmidt, "IP4 to IPv6: Challenges, solutions and lessons," *Telecommunications Policy, vol. 38*, 2014.

[89] J. Roberts and N. Rapp. Buterin and V. Griffith, "Exclusive: Nearly 4 Million Bitcoins Lost Forever, New Study Says. Fortune," http://fortune.com/2017/11/25/lost-bitcoins/, Jun 2018.

[90] P. Wilson, R. Plzak and A. Pawli, "IPv6 Address Space Management," *RIPE-343*, Jun 2018.

[91] B. Carpenter, R. Atkinson and H. Flinck, "Renumbering Still Needs Work," IETF RFC 5887, 2010.

[92] "Provable Documentation," http://docs.provable.xyz/#home, accessed: 2020-02-6.

[93] D. Kessens, T. J. Bates, C. Alaettinoglu, D. Meyer, C. Villamizar, M. Terpstra, D. Karrenberg, and E. P. Gerich, "Routing Policy Specification Language (RPSL)," RFC 2622, Jun. 1999. [Online]. Available: https://rfc-editor.org/rfc/rfc2622.txt

[94] R. Bush and R. Austein, "The Resource Public Key Infrastructure (RPKI) to Router Protocol, Version 1," IETF RFC 8210, 2017.

[95] T. Bruijnzeels, O. Muravskiy, B. Weber, Cobenian and R. Austein, "The RPKI Repository Delta Protocol (RRDP)," IETF RFC 8182, 2017.

[96] A. Bahga and V. Madisetti, *Blockchain Applications: A Hands-On Approach.* VPT, 2017.

[97] M. Wohrer and U. Zdun, "Smart contracts: security patterns in the ethereum ecosystem and solidity," in *2018 International Workshop on Blockchain Oriented Software Engineering*, March 2018, pp. 2–8.

[98] "IPv6 Address Space Management," https://www.parity.io/what-is-a-light-client/, accessed: 2020-02-4.

[99] Etherscan, "Ethereum GasPrice History," https://etherscan.io/chart/gasprice, March 2019.

[100] V. Buterin, "Gas cost changes for IO-heavy operations," https://github.com/ethereum/EIPs/blob/master/EIPS/eip-150.md, Sep 2016.

[101] S. Angieri, A. García-Martínez, B. Liu, Z. Yan, C. Wang, and M. Bagnulo, "A distributed autonomous organization for Internet address management," *IEEE Transactions on Engineering Management*, 2019.

[102] Andree Toonk, "Masive route leak causes Internet slowdown," https://www.bgpmon.net/massive-route-leak-cause-internet-slowdown/, Jun 2015.

[103] J. Fleury et. al., "Anatomy of a route leak," https://www.lacnic.net/innovaportal/file/4016/1/cloudflare---anatomy-of-a-route-leak.pdf, 2019.

[104] P. Faratin, D. D. Clark, S. Bauer, W. Lehr, P. W. Gilmore, and A. Berger, "The growing complexity of internet interconnection," *Communications & strategies*, no. 72, p. 51, 2008.

[105] F. V. Silveira, "Rpki repositories and the ripe database in the cloud," https://labs.ripe.net/author/felipe_victolla_silveira/rpki-repositories-and-the-ripe-database-in-the-cloud/, [Online; accessed 9-Jul-2021].

[106] F. V. Silveira, K. Ranjbar, D. Karrenberg, F. Cunningham, V. Manojlovic, and A. Gollan, "Ripe ncc and the cloud: Let's start again," https://labs.ripe.net/author/felipe_victolla_silveira/ripe-ncc-and-the-cloud-lets-start-again/, [Online; accessed 9-Jul-2021].

[107] H. Cummunity, "Hyperledger fabric," https://www.hyperledger.org/use/fabric, [Online; accessed 30-Jun-2021].

[108] J. Ma, Y. Jo, and C. Park, "Peerbft: Making hyperledger fabric's ordering service withstand byzantine faults," *IEEE Access*, vol. 8, pp. 217 255–217 267, 2020.

[109] J. Sousa, A. Bessani, and M. 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)*, 2018, pp. 51–58.

[110] D. C. W. L. Jr., K. Seo, and S. Kent, "X.509 Extensions for IP Addresses and AS Identifiers," RFC 3779, Jun. 2004. [Online]. Available: https://rfc-editor.org/rfc/rfc3779.txt

[111] H. Community, "Query the couchdb state database with pagination," https://hyperledger-fabric.readthedocs.io/en/release-2.2/couchdb_tutorial.html#cdb-pagination, [Online; accessed 9-Jul-2021].

[112] S. Angieri, "Irb_blockchain," https://github.com/steang91/IRB_Blockchain, [Online; accessed 9-Jul-2021].

[113] V. Giotsas, M. Luckie, B. Huffaker, and k. claffy, "Inferring Complex AS Relationships," in *ACM Internet Measurement Conference (IMC)*, Nov 2014, pp. 23–30.

[114] IANA, "Autonomous system (as) numbers," https://www.iana.org/assignments/as-numbers/as-numbers.xhtml, [Online; accessed 6-Jul-2021].

[115] L. Gao, "On inferring autonomous system relationships in the internet," *IEEE/ACM Transactions on Networking*, vol. 9, no. 6, pp. 733–745, 2001.

[116] R. Bush and R. Austein, "The Resource Public Key Infrastructure (RPKI) to Router Protocol, Version 1," RFC 8210, Sep. 2017. [Online]. Available: https://rfc-editor.org/rfc/rfc8210.txt

[117] D. Meyer, C. Alaettinoglu, D. C. Orange, M. R. Prior, and D. J. Schmitz, "Using RPSL in Practice," RFC 2650, Aug. 1999. [Online]. Available: https://rfc-editor.org/rfc/rfc2650.txt

[118] B. Kuerbis and M. Mueller, "Internet routing registries, data governance, and security," *Journal of Cyber Policy*, vol. 2, no. 1, pp. 64–81, 2017.

[119] M. Luckie, B. Huffaker, A. Dhamdhere, V. Giotsas, and K. Claffy, "As relationships, customer cones, and validation," in *Proceedings of the 2013 conference on Internet measurement conference*, 2013, pp. 243–256.

[120] V. Giotsas and S. Zhou, "Valley-free violation in internet routing - analysis based on bgp community data," 06 2012, pp. 1193–1197.

[121] J. Brenes, G.-M. Alberto, M. Bagnulo, A. Lutu, and C. Pelsser, "Power prefixes prioritization for smarter bgp reconvergence," *IEEE/ACM Transactions on Networking*, vol. 28, no. 3, pp. 1074–1087, 04 2020.

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

[123] U. D. Team, "Unstoppable domains," https://unstoppabledomains.com/, [Online; accessed 19-Oct-2021].

[124] "IPv6 Address Space," https://www.iana.org/assignments/ipv6-address-space/ipv6-address-space.xhtml, May 2018.