Blockchain-based Public Key Infrastructure for Inter-Domain Secure Routing

Alfonso de La Rocha Gómez-Arevalillo, Panos Papadimitratos

▶ **To cite this version:**

Alfonso de La Rocha Gómez-Arevalillo, Panos Papadimitratos. Blockchain-based Public Key Infrastructure for Inter-Domain Secure Routing. International Workshop on Open Problems in Network Security (iNetSec), May 2017, Rome, Italy. pp.20-38. hal-01684192

HAL Id: hal-01684192

https://inria.hal.science/hal-01684192v1

Submitted on 15 Jan 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Blockchain-based Public Key Infrastructure for Inter-Domain Secure Routing

Alfonso de la Rocha Gómez-Arevalillo and Panos Papadimitratos

Networked Systems Security Group
KTH Royal Institute of Technology
www.ee.kth.se/nss

Abstract. A gamut of secure inter-domain routing protocols has been proposed in the literature. They use traditional PGP-like and centralized Public Key Infrastructures for trust management. In this paper, we propose our alternative approach for managing security associations, *Secure Blockchain Trust Management (SBTM)*, a trust management system that instantiates a blockchain-based PKI for the operation of secure routing protocols. A main motivation for SBTM is to facilitate gradual deployment across Autonomous Systems (ASes) with minimal operational and economical costs. An initial performance evaluation supports the practicality of SBTM and outlines potential benefits of blockchain-based frameworks for securing routing protocols.

Keywords: trust management system, blockchain protocol, secure routing

1 Introduction

Secure inter-domain protocols safeguard the discovery of communication paths among Autonomous Systems (ASes) from intentional attacks and misconfigurations; for example, the announcement of unauthorized prefixes or the outright impersonation of ASes, or the modification of communication. Currently, the Internet relies on the Border Gateway Protocol (BGP) [14], a versatile protocol with no security guarantees. Research on secure inter-domain protocols, in particular on securing BGP, focuses on addressing security weaknesses in different ways, e.g., by providing integrity, confidentiality, authentication, and authorization for routing messages [13],[10]. Research efforts by standardization bodies, industry and academia have resulted in the development numerous proposals to secure BGP [5], including four representative ones: *Secure BGP (S-BGP)* [12], *Secure origin BGP (SoBGP)* [16], *Interdomain Route Validation (IRV)* [9] and *Path-end validation* [7]. Each of them relies on its own approach for the management of security associations, keys, credentials and identities required for the secure protocol operation. The deployment of secure routing systems has faced significant difficulties due to the complexity and cost of managing trust in large-scale networked systems.

Consequently, we present in this paper *Secure Blockchain (SBTM)*, a general and flexible trust management framework for inter-domain routing based on a blockchain protocol. SBTM provides the establishment of security associations, the management of routing entities' identities, credentials, keys and prefix ownerships, for any secure inter-domain routing system. This is achieved by replacing the current Public Key Infrastructure (PKI) or web of trust based approaches with a blockchain protocol. All relevant security information and credentials, for all entities involved in the secure routing protocol, are to be stored in the blockchain, making it available for every (border) router of the system through an SBTM-wallet. When an AS provides a path attestation or a prefix, according to the specifics of the routing protocol, each receiving router can authenticate the message and validate if the announcing AS was authorized to perform this task, by inspecting the security information (public keys, protocol-specific certificates, etc.) stored in the blockchain.

SBTM can provide benefits over existing approaches for trust management for secure routing protocols. The use of a blockchain makes the system fully *decentralized*, removing the existence of central points of failure (Certificate Authorities (CAs)) and the cumbersome establishment of trust with remote, in terms of trust, ASes in web-of-trust approaches. The deployment and operation of SBTM requires minimal additions to any AS: (i) installation of the SBTM-wallet in each of its border routers (so they can access all the information stored in the blockchain), and (ii) deployment of some additional devices to "mine" the blockchain, i.e., earn the crypto-currency required to perform operations over the blockchain. Finally, SBTM inherits the safe-guard properties of a proof-of-work based blockchain, making changes over the information stored in the blockchain untractable as a change in the information registered in the blockchain will require an attacker to modify consequently all prior transactions in the public ledger (as all transaction in the blockchain depends from all previous transactions).

The rest of the paper is organized as follows: Sec. 2 surveys secure routing frameworks and current trust management proposals; Sec. 3 provides a system and Sec. 4 requirements; Sec. 5 describes in detail SBTM, illustrates deployment and operation with specific secure routing protocols in mind; Sec. 6 provides a brief analysis, followed by a preliminary performance evaluation (Sec. 7) before we conclude.

2 Related Work

As reported in the literature, several attacks on BGP resulted in instabilities and outages for entire Internet domains [2], intentional blocking of web sites [3], illegal seizure of routing paths or IP prefixes to snoop traffic [4], etc. To avoid such attacks, four representative proposals, already mentioned, are: *Secure BGP (S-BGP)* [12], *Secure origin BGP (SoBGP)* [16], *Interdomain Route Validation (IRV)*, [9] and *Path-end validation* [7]. Each of these proposals addresses BGP security in a different way, seeking to provide properties, such as path and origin

authentication, to ensure an advertised path is valid and that the announcement originated at an authorized entity. Each framework relies on its own approach to manage trust.

S-BGP relies on two PKIs to provide all information related to AS certificates, identities and prefix ownerships. With certificates binding public keys to identities and attributes, digital signatures on secure routing protocol messages can be validated. Path-end validation does not change BGP and performs origin authentication through the use of RPKI [15], a special PKI that binds an IP-prefix with the number and public key of the Autonomous System (AS) that owns it. So-BGP, on the other hand, performs origin and path/topology authentication based on three types of certificates that are cross-signed by ASes, without making use of a central authority, in a web-of-trust (also known as PGP-like) approach: *EntityCerts* bind a public key to each SoBGP-speaking router, *PolicyCerts* provide details on policy, route requirements or configure protocol parameters, and *AuthCerts*, provide address ownerships and delegation. So-BGP removes the need for the (one or two) PKIs trading off the achieved security. IRV has no additional certificates but it relies on an independent security infrastructure border routers can request verification of UPDATE messages from. This validation infrastructure is orthogonal to the routing protocol and relies on having each AS running its own validation server and responding to queries.

Our objective here is not to propose any new secure routing protocol or evaluate their relative security. Rather, we are after a flexible, decentralized, trust management approach that is at once secure and practical, amenable to incremental deployment. For that purpose, we leverage blockchain technologies and in particular Certcoin [8], a decentralized PKI based on a blockchain protocol. Our SBTM proposal provides an alternative **trust management system** that can complement any secure routing protocol: simply put, practically all certificates are generated by ASes and they become part of the blockchain, made available for any other router to lookup and validate. The objective is two-fold: decentralized functionality and gradual deployment, with addition of ASes and routers within ASes, along with strong security, thanks to the blockchain, and minimal involvement of the Internet governing body.

3 System Model

A generic system model, with notation summarized in Table 1, faciliates the presentation of SBTM. Inter-domain routing systems (ρ) discover paths connecting any two Autonomous Systems (ASes) of a network, N, through the exchange of specialized messages. An AS is a large collection of connected network devices and routing prefixes, $IP_{AS_{id}}$, under the control of a common administrator (e.g., an ISP) with defined routing policies. A central authority, such as IANA (Internet Assigned Numbers Authority), is in charge of recognizing a network entity as an Autonomous System, providing it with an AS identification number, AS_{id}, along with a set of IP address prefixes, $IPs_{AS_{id}}$, for its subnetworks, $N_{AS_{id}}$.

Table 1. Secure routing systems notation

Symbol	Definition
ρ	Inter-domain routing system
ϕ_s	Secure routing protocol
ϕ_{ns}	Insecure routing protocol
N	Network
AS_{id}	AS id
$IP_{AS_{id}}$	IP prefixes owned by AS_{id}
R_i	Border router of AS_i
τ	Trust management system
M_{ij}	Routing message from AS_i to AS_j
$P_{AS_{id}}$	AS public key or certificate
$S_{AS_{id}}$	AS private key
$f_\tau(...)$	Identity binding function issued by τ
$\gamma_i = f_\tau(AS_{id}, P_{AS_{id}})$	Public key binding AS_i
$\mu_i = f_\tau(AS_{id}, IPs_{AS_{id}})$	IP ownership AS_i
$\lambda_i = f_\tau(AS_{id}, IPs_{prefix}, route, topology...)$	Protocol-specific info AS_i

Internet routing is hierarchical thus, every AS_x has two types of network devices: *internal routers*, running intra-domain routing protocols, and *border routers*, R_x, connected to other ASes' border routers interconnecting their networks. Inter-domain routing protocols set up paths from an AS_i, to any other AS_x through the exchange of routing messages, M_{ix}. A routing system may run an *insecure inter-domain routing protocol,* ϕ_{ns}, which performs path discovery between ASes without any security features. *Secure inter-domain routing protocols (ϕ_s)* can represent the secure variant of a non-secure routing protocol ϕ_{ns}, or a standalone secure routing protocol.

A secure routing protocol, ϕ_s, needs a trust management system, τ, that enables the establishment of security associations between routers and ASes. Thus, τ manages all security information related to ASes and the secure routing protocol. Let a general function, $f_\tau(...)$, bind the AS identity to the related security information, such as public keys, IP ownership, etc.; the specific implementation of $f_\tau(...)$ depends on τ.

In current τ instances, the output of $f_\tau()$ is essentially a signed certificate. Without loss of generality, we consider

$$f_\tau(Sec.info) = Cert_\tau = \{Sec.info, sig_\tau(Sec.info)\}$$

where $Sec.info$ represents all the security information included in the certificate, according to the certificate type, and $sig_\tau(Sec.info)$ represents a digital signature. sig_τ may represent a single signature from a single trusted party (a CA in a PKI) or a vector of signatures from different entities of the trust management infrastructure, i.e., a trusted path (such as a hierarchy of CAs or a set of trusted ASes in a PGP-like protocol).

In Certcoin and other blockchain-based trust management systems, the use of certificates is not needed, as the binding of identities and their related secu-

rity informations are directly performed by including, and conveniently signing, identities and their related security information in a transaction. Thus, $f_\tau()$ is:

$$f_\tau(Sec.info) = Transact.\{AS_{id}, Sec.info, sig_\tau(AS_{id}, Sec.info)\}$$

τ will manage the binding between an AS identity and its public key, $\gamma_{AS_{id}}$, so that every member of the system can validate messages signed by the corresponding AS secret key, $S_{AS_{id}}$; the binding of an AS identity and its prefix ownership, $\mu_{AS_{id}}$; and the generation, management and signing of any other security information required by the security protocol for its specific operation, $\lambda_{AS_{id}}$. These protocol-specific certificates may include knowledge related to the system topology, policies, IP advertisement delegations, IP ownerships, etc. Some examples of $\lambda_{AS_{id}}$ from the already mentioned secure inter-domain routing protocols would be: IP advertisement delegations in S-BGP; *EntityCerts*, *PolicyCerts* and *AuthCerts* in So-BGP; IP ownership certificates in RPKI or IRV-responses.

Based on the description above, an AS_i may initially be represented as:

$$AS_i = \{AS_{id}, IP_{AS_{id}}, R_i^1, ..., R_i^n, \gamma_{AS_{id}}\}$$

τ is the entity in charge of managing and signing the following tuple of certificates for the AS_i: $\{\gamma_{AS_{id}}, \mu_{AS_{id}}, \lambda_{AS_{id}}\}$

We considered two type of adversaries: External adversaries, network entities not participating in the routing protocol, without valid credentials from the trust management system. These can disrupt the operation of the system by intercepting packets, replaying and modifying messages and forging messages; and internal adversaries with valid credentials in the system and with access to the trust management infrastructure. These adversaries have access to the blockchain and are able to perform transactions in it, thus being able to store new information in the blockchain.

4 Requirements

The aim is to deploy τ along with a secure routing protocol ϕ_s. We are interested in requirements for τ independently of the correctness of ϕ_s. We are aware that the design of ϕ_s will influence τ in terms of the information included in the λ and μ certificates. In other words, τ should provide all the security associations needed for the correct operation of ϕ_s.

Requirement 1: *If ϕ_s, enabled by some τ', is correct and ensures a set of security properties, then ϕ_s, enabled by some $\tau \neq \tau'$, should also be correct and ensure the same security properties.*

For example, if two border routers R_i and R_j from two different ASes rely on ϕ_s over τ' to run a secure routing protocol which ensures origin, topology and path authentication, the replacement of the ϕ_s trust infrastructure from τ' to τ should result in a system that satisfies the same security properties. In other words, τ should be able to "deliver" ϕ_s all information needed and provided by τ'.

Requirement 2: *τ should provide the facilities for every AS_i, and therefore every $R_j \in AS_i$, to obtain all the λ, γ, μ certificates required to authenticate and validate ϕ_s routing messages or any other of ϕ_s security mechanisms.*

Let $R_x \in AS_x$ that sends a routing message to $R_i \in AS_i$. R_i must be able, through simple request to τ, to: (a) get the public key of R_x to authenticate the message, (b) get the IP prefixes owned by R_x and its address attestations to validate that it had permission to advertise the path included in the routing message, (c) validate that the message was not modified or injected in the system.

Requirement 3: *τ should enable a gradual, incremental deployment, from an insecure routing protocol ϕ_{ns} to its secure version ϕ_s. This should (i) cause no changes in the underlying protocols, (ii) allow for the temporary coexistence of ϕ_{ns} and ϕ_s, and (iii) support ϕ_s as per Req. 1 and 2.*

Any border router R_i from AS_i should be able to validate ϕ_s routing messages, M_{ji}, from AS_j even if they traverse networks still running ϕ_{ns}, and vice versa. ϕ_s and ϕ_{ns} should be able to coexist in ρ, allowing the exchange of routing messages M_{ix}, M_{xi} between border routers $R_i \in AS_i$ running ϕ_s and $R_x \in AS_x$ running ϕ_{ns}.

Requirement 4: *τ should allow a seamless join of ASes and routers to ρ.*

If router $R_i \in AS_i$ is to join the system, this should be done solely by performing a blockchain transaction. Identically, the only operational cost for a new AS_j, and its border routers $R_j \in AS_j$, joining the system should be the generation of new certificates by τ, with no modification or changes over ϕ_s. Moreover, additional joins of numerous routers and ASes should not penalize the response time of operations requested by entities of the routing system over τ.

Requirement 5: *τ should offer the capability to quickly react against the compromise of an AS credentials and it should protect the integrity of the security information (λ_i, μ_i and γ_i certificates).*

For example, if a border router $R_i \in AS_i$ is hacked and its keys are compromised, τ should allow for quick revocation of the compromised keys and renewal of keys and certificates $(\lambda_i, \mu_i, \gamma_i)$, to contain, as much as possible, potential harm and disruptions. Ideally, τ should enable detection of the malicious behavior after the compromise. Moreover, no external adversaries without valid credentials should be able to modify or forge the information stored in the trust management system.

5 Secure Blockchain Trust Management (SBTM) system

5.1 SBTM overview

SBTM is a decentralized trust management infrastructure, τ, for secure inter-domain routing protocols, ϕ_s, providing every border router $R_i \in AS_i \in \rho$ with the tools and capabilities to retrieve all the security information and certificates for any other $AS_j \in \rho$ in order to validate ϕ_s routing messages. SBTM operates in the following stages:

Recognizing an AS: An AS_i can be described by the following tuple:

$$AS_i = \{AS_{id}, IPs_{AS_{id}}, R_i^1, ..., R_i^n, \gamma_{AS_{id}}\}$$

AS_i must have been recognized as such by IANA, the central authority in charge of managing the Internet and BGP. The central authority provides AS_i with its AS_{id}, a set of owned IP prefixes, $IPsAS_i$, an *SBTM-wallet*, and all the security information stored in γ_i and μ_i. All this information is linked to the SBTM-wallet of the AS_i.

We borrow the concept of wallet from the crypto-currency world. SBTM-wallets are similar to those in Bitcoin. They are a piece of trusted software in charge of: (i) storing the identities and necessary keys for an AS to participate in the blockchain protocol, (ii) allowing entities to inspect and perform operations over the blockchain, (iii) storing the amount of currency, owned by an AS, required to perform operations over the blockchain.

SBTM-wallets implement a secure login system; even if an adversary compromises a network device running a SBTM wallet, the AS's sys-admin credentials would be needed to access and operate the SBTM-wallet on behalf of the AS. Strong security measures already used in corporate environments could be applied to enhance the security of this piece of software. Once the central authority have recognized AS_i as such, AS_i is provided with a link to a trusted repository where downloads its SBTM-wallet.

AS initial registration to the blockchain: With the SBTM-wallet, AS_i generates its initial online and offline credentials, connected with a pair of online and offline public and private keys linked to its identity and its SBTM-wallet. Once created, these credentials are automatically announced to the central authority SBTM-wallet, which is the one responsible for issuing the *register* operation in the blockchain, appropriately signed, including the certificates and security information, $AS_i = \{AS_i, IPs_{AS_i}, \gamma_{AS_i}, \mu_{AS_i}\}$. After this transaction is validated, the AS_i identity becomes publicly available to all other AS_j running ϕ_s in *rho*. All these initial transactions are free of charge (incur no cost).

Fig. 1 illustrates a set of AS_i border routers with their SBTM-wallets installed. Neighboring R_i are connected over network links, across which ϕ_s routing messages are exchanged. Moreover, SBTM-wallets are continuously communicating with each other and with the blockchain through a P2P protocol across the Internet.

The blockchain represents a public ledger with a set of validated transactions containing information related to AS identity, credentials, and operation, available to every member of the blockchain protocol. This public ledger, stored in every device holding a wallet is periodically updated to include newly validated transactions. As proposed in Certcoin, different storage techniques may be considered for trusted devices to jointly store the blockchain. This is out of the scope of this paper.

Installation and registration of specific security information: Consider a set of ASes that joined the system and installed their SBTM-wallets to their border routers, as illustrated in Fig. 2. Every SBTM-wallet in the system

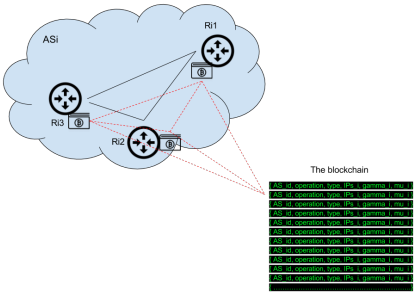

Fig. 1. Communication between $R_i^x \in AS_i$ and the blockchain

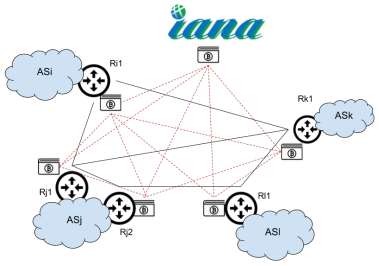

Fig. 2. Architecture of a routing system over SBTM

has full access to the blockchain and they all form a mesh network connected to the central authority SBTM-wallet.

ASes have to register their protocol-specific security information in the blockchain in order to make it available for other entities in ρ running ϕ_s. Let AS_i that wishes to register a protocol-specific λ_i so that this information can be validated by any other entity of ρ, e.g., AS_j. Then, AS_i sends AS_j its certificate λ_i. AS_j will then validate it, sign it, and perform an *update* transaction in the blockchain to make λ_i available for everyone. To make this *update* transaction, AS_j will lookup AS_i's last validated transaction in the blockchain and it will update this transaction, by reissuing it, including λ_i and its signature. The cost of transactions for the registration of protocol-specific information in the blockchain is taken over by the AS whose information is being modified, AS_i in this scenario. Blockchain protocols allow the implementation of additional intelligence for transactions (even if it is not a smart-contract blockchain); conditions can be implemented to transactions so that, for instance, a transaction is not validated until a specific entity involved in the transaction, and not its issuer, pays the cost. Thus, in SBTM, the transaction for protcol information registration is not validated in the blockchain until AS_i pays for the cost. More details of this process will be presented in the next subsection.

Different ϕ_s may present different types of certificates or security information in λ_i with different signature and validation policies, requiring SBTM to be configured appropriately. For example, in a soBGP/PGP-like ϕ_s a λ_i would just need to be signed by some trusted entities in the system to be valid, as AS_j in the previous case. However, in a centralized PKI based ϕ_s, this λ_i may need to be transmitted, signed and updated in the blockchain by a central authority, such as IANA. The cost of the protocol-specific security information registering transaction, even if its performed by a validating third-party such as AS_j or $IANA$, will be taken over by the entity for which the protocol information is registered, AS_i. We will outline example configurations of SBTM for actual ϕ_s below.

Up to this point, every router $R_i \in AS_i$ uses the same identity γ_{AS_i}. However, AS_i can assign a different identity, bound to a public key, for each of its routers providing router-level identification, authentication and non-repudiation. Thus, AS_i performs an *update* transaction, an appropriately signed tuple (R_i, P_{Ri}) for each of its border routers, where R_i is the id for AS_i's router and P_{Ri} its corresponding public key.

Mining the blockchain: For correct operation and consistency of the blockchain, every AS that runs SBTM mines the blockchain. These computations can be done in each border router or in dedicated servers owned by the AS. Mining is necessary in every crypto-currency protocol in order to validate new transactions in the blockchain. The miners try solving a hard computational problem. When a miner finds a solution, some currency is added to its wallet, in our case Certcoins. Simply put, ASes need to have enough crypto-currency (Certcoins) for basic operations in the blockchain. The central authority also needs resources for mining the blockchain and avoid potential internal attacks and verify issued transactions.

SBTM operation within ϕ_s: Once an AS is recognized as member of the system, registered its protocol-specific certificates and installed its SBTM-wallet, it is ready to run seamlessly ϕ_s based on its new SBTM-based trust management system.

When AS_i receives a ϕ_s routing message from AS_j, it performs a lookup in the blockchain to find the last valid transaction over AS_j. Such a transaction gives AS_i a tuple $(AS_j, \gamma_j, \mu_j, \lambda_j)$ that is all the security information required to validate routing messages from AS_j. AS_i has to perform the same look up in the blockchain as every message arriving from an unknown $AS_x \in \phi_s \in \rho$. This process is seemingly computationally expensive, however, there are many ways of bypassing or minimizing the cost of lookups: e.g., caching information about frequently seen ASes, or the definition of specific look up policies where an intermediate system is used to retrieve updated identities and communicate them to every router (relieving routers from this task).

New border routers or ASes joining the system: If a new AS_m joins ρ running ϕ_s, it has to follow the same steps as every AS that previously joined ρ. This process is transparent for entities already in the system. When new routing messages arrive from AS_m, some AS_i just does a lookup in the blockchain to verify the information required to validate any routing message from AS_m.

5.2 SBTM functionality: details and illustration of ϕ_s specifics

SBTM is orthogonal to the secure routing protocol ϕ_s. However, the ϕ_s deployed over SBTM needs protocol-specific configuration. Thus, to better illustrate SBTM operation, we describe below SBTM steps along with specific actions when used for a *PGP-like* trust ϕ_s such as soBGP [16], and one that uses a *centralized* trust ϕ_s as S-BGP [11].

Initial registration to the system: AS_i receives its SBTM-wallet for the first time and generates an online asymmetric key pair $(P^{on}_{AS_i}, S^{on}_{AS_i})$ and an offline

asymmetric key pair $(P^{off}_{AS_i}, S^{off}_{AS_i})$. Every AS, through its wallet, is limited to the generation of a single online pair and a single offline pair. This is not the case in Bitcoin or Certcoin wallets, where an infinite number of identities may be generated for a single user. Once the online and offline pairs are generated, the wallet communicates the public keys of both identities encrypted to *IANA*'s wallet, which is in charge of registering these identities in the blockchain (along with the AS's assigned prefixes). *IANA*'s wallet registers the following performs transactions in the blockchain for each AS:

$$(AS_{id}, register, online, P^{on}_{AS_{id}}, IPs_{AS_{id}}, sign_{IANA}\{AS_{id}, P^{on}_{AS_{id}}, IPs_{AS_{id}}\})$$

$$(AS_{id}, register, offline, P^{off}_{AS_{id}}, sign_{IANA}\{AS_{id}, P^{off}_{AS_{id}}\})$$

AS_{id} is the AS number, $P^{on}_{AS_{id}}$ and $P^{off}_{AS_{id}}$ are the online and offline public keys for the online and offline identity, respectively, $sign_{IANA}\{AS_{id}, P^{on}_{AS_{id}}, IPs_{AS_{id}}\}$ is a signature computed with *IANA*'s wallet online secret key, S^{on}_{IANA}; using its corresponding public key, every router can verify that the registration of the identities and the address attestations are valid. *IANA*'s public key is available to every router of the system. $IPs_{AS_{id}}$ are the IP prefixes assigned by *IANA* to this AS. ASes store the private key for the online identity directly inside the SBTM-wallet, while the offline private key should be securely stored in an offline device. The offline identity is used for revocation and modification operations over the online and offline identities. *IANA*'s wallet is the only entity allowed to make *register* operations in the blockchain. This may be easily configured in Certcoin by making the registration operation free for *IANA*'s wallet and with a monetary cost of infinite Certcoins for the rest of users in the system entities. Moreover, because of this registration scheme, IANA is able to control and avoid the registration of duplicate online or offline identities or, even, the registration of illegal prefixes for any AS.

Installation and registration of protocol-specific security information: ASes make protocol-specific information λ available in the blockchain through conveniently signed *update* operations, in order to start using ϕ_s. The specific signature scheme for λ will depend on ϕ_s. Let us assume a PGP-based ϕ_s. In order for AS_i to publish its λ_i in the blockchain, the latter has to be cross-signed by any other trusted AS_j in the routing system. Therefore, AS_i will send its certificate to AS_j which will add the certificate in the blockchain on behalf on AS_i

Thus, AS_j will look up in the blockchain the last *register*, *upload* or *revoke* transaction performed over AS_i and update it, by reissuing a new transaction, with AS_i's updated λ_i certificate. This way, AS_j validates AS_i's λ_i, while any other AS can look up AS_i public key and identity in the blockchain. Assuming that AS_j finds that the last operation performed to AS_i was its initial identity registration, $(AS_i, register, online, P^{on}_{AS_i}, IPs_{AS_i}, sign_{IANA}\{AS_i, P^{on}_{AS_i}, IPs_{AS_i}\})$, then AS_j will reissue this transaction including λ_i and its signature at the end of the transaction. The type of the transaction will change from its original type

to *update*. Thus, the original transaction was a *register* operation, and the new *update* transaction including λ_i will be:

$$(AS_i, update, online, P^{on}_{AS_i}, IPs_{AS_i}, \\ sign_{IANA}\{AS_i, P^{on}_{AS_i}, IPs_{AS_i}\}, \lambda_i, sign_{AS_j}\{AS_i, \lambda_i\})$$

This transaction will not be validated until AS_i pays the fee for the transaction. As already mentioned before, the AS whose protocol information is modified in the blockchain is the one responsible for its cost, even though another AS_j performed the transaction on its behalf.

If, for instance, AS_i also wishes to upload a set of router-specific identities for routers R_{i1} and R_{i2}, in order to enable router-level authentication and non-repudiation in the system, a new *update* transaction is performed over its own identity and signed and paid by itself, including each router identity and related public key:

$$(AS_i, update, online, P^{on}_{AS_i}, IPs_{AS_i}, sign_{IANA}\{AS_i, P^{on}_{AS_i}, IPs_{AS_i}\}, \\ \{(R_{i1}, P_{Ri1}), (R_{ix}, P_{Rix})\}, sign_{AS_i}\{AS_i, (R_{i1}, P_{Ri1}), (R_{ix}, P_{Rix})\}, \\ \lambda_i, sign_{AS_j}\{AS_i, \lambda_i\})$$

The installation of router-specific identities and protocol-specific security information may be performed in the same *update* transaction. Fig.3 summarizes information included in a single transaction; The specific cross-signing method used for this operation may differ according to the specific ϕ_s.

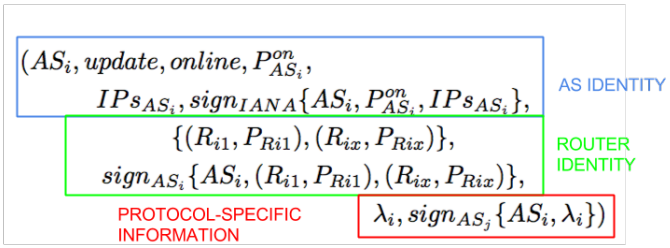

Fig. 3. Information included in a SBTM transaction

soBGP installation of soBGP certificates: The protocol-specific security information is stored in the *EnityCert*, *PolicyCert* and *AuthCert*. Thus, an AS_i wishing to publish in the blockchain its soBGP certificates follows the general procedure presented above. AS_i sends the three *Certs* to AS_j, or a set of AS_j, in charge of validating the information included in them. AS_j finds the last transaction performed over AS_i in the blockchain and updates it to include the

soBGP certificates (briefly denoted *xCerts*) as follows:

$$(AS_i, update, online, P^{on}_{AS_i}, IPs_{AS_i}, sign_{IANA}\{AS_i, P^{on}_{AS_i}, IPs_{AS_i}\}, \\ xCerts_{AS_{id}}, sign_{AS_j}\{AS_i, xCerts_{AS_{id}}\})$$

S-BGP address attestation: Address attestations are protocol-specific security information that must be registered in the blockchain. Suppose that AS_i owns prefixes IPs_{AS_i}, and whishes AS_j to also be able to advertise these prefixes. Thus, AS_i looks up in the blockchain the last *register*, *update* or *revoke* transaction performed over AS_j and updates it to allow AS_i to advertise the prefixes. Let the last operation performed over AS_j is

$$(AS_j, register, online, P^{on}_{AS_j}, IPs_{AS_j}, sign_{IANA}\{AS_j, P^{on}_{AS_j}, IPs_{AS_j}\}) \ ,$$

AS_i then adds the prefixes that it wants AS_j to advertise to this transaction, updating the transaction type from its original value, *register* in the current example, to an *update*:

$$(AS_j, update, online, P^{on}_{AS_j}, IPs_{AS_j}, sign_{IANA}\{AS_j, P^{on}_{AS_j}, IPs_{AS_j}, \\ IPs_{AS_i}, sign_{AS_i}\{AS_j, IPs_{AS_i}\})$$

This way, when AS_j advertises a prefix owned by AS_i, other ASes can verify the advertisement was allowed by AS_i by $sign_{AS_i}\{AS_j, IPs_{AS_i}\}$ using AS_i online public key. The cost for this transaction is paid by the AS reponsible for the address attestation, AS_i due to the paying conditions defined for every transaction in SBTM.

Managing protocol-specific security information: If AS_j wants to remove its signature over AS_i's λ_i, to make it invalid or to force other members of the system to cross-sign λ_i in order to re-verify it, a *revoke* transaction will be performed as follows:

$$(AS_i, revoke, online, P^{on}_{AS_i}, IPs_{AS_i}, sign_{IANA}\{AS_i, P^{on}_{AS_i}, IPs_{AS_i}\}, \\ sign_{AS_j}\{AS_j, sign_{AS_j}\{AS_i, \lambda_i\}\})$$

AS_j would find the last transaction over AS_i, it would remove the certificate λ_i and will include it at the end of its transaction a signature of its previous signature in the *update* transaction, and its own id, AS_j, so that the rest of the members in the system can recognize him as the one that removed λ_i from AS_i. The cost for this transaction is taken over by AS_j. Additionally, removals of router-specific identities would be performed the same way as updates of protocol-specific information. If instead of completely removing protocol-specific or router-specific information, an AS only wants to update this information, the type of transaction to be performed will be an *update* transaction, instead of a *revoke* transaction.

Revoke transactions used to delete protocol-specific certificates of an AS (not if they are used to revoke keys) add a signature $sign_{AS_j}\{AS_j, sign_{AS_j}\{AS_i, \lambda_i\}\}$

at the end of the transaction to indicate that AS_i certificates were removed. When another AS_k finds this operation, it may decide to check the previous AS_i transaction to validate that, indeed, AS_j only modified in the *revoke* transaction or certificates signed by it, as it was entitled to. The next *update* operation over AS_i will remove this signature as the λ_i removal after this is considered verified and new λ_i information may be added normally in an *update* operation. As every transaction is recorded in the blockchain, if any anomaly or illegal information not compliant with ASes policies is detected by the involved AS, or any other entity of the system, this will immediately be reported to *IANA* for its subsequent fix.

One may immediately think that an easy attack by a recognized AS (internal adversary) would be to remove a set of valid λ certificates for an specific AS through the generation of a subsequent set of *update* or *revoke* transactions, destabilizing this AS operation in the routing system. However, this is solved by the blockchain itself. Every transaction is registered in the blockchain so, periodic validations through the whole blockchain may easily be performed by miners, *IANA* or even double-checked by the involved ASes in order to avoid this kind of misbehavior. *IANA* can periodically monitor the blockchain to detect malicious operations, and ASes can double-check a transaction, by performing an additional look up when facing a *revoke* or *update* transaction to verify that no illegal transactions were performed.

Consider an entity with a valid AS with a valid SBTM-wallet software given by *IANA* and with enough Certcoins to post requests into the blockchain. Transactions should be correct not to be discarded by miners or *IANA* during their verification. As they come signed, attacks can be tracked with a low impact for the system, due to the inherent traceability properties of blockchain protocols. Removals of every protocol-specific identities for an AS would be performed using the same *revoke* operation used to remove protocol-specific information from a transaction. We illustrate this process of removal of protocol-specific certificates in S-BGP and soBGP.

Removal or update of soBGP certificates: The removal or update of soBGP *Certs* is performed following the general procedure explained above, replacing λ_i in the signature by the specific *Certs* that want to be removed or updated.

S-BGP removal or update of address attestation: If AS_i does not want AS_j to advertise IPs_{AS_i} anymore it posts a revoke transaction:

$$(AS_j, revoke, online, P^{on}_{AS_j}, IPs_{AS_j}, sign_{IANA}\{AS_j, P^{on}_{AS_j}, IPs_{AS_j}\}, \\ sign_{AS_i}\{AS_i, sign_{AS_i}\{AS_j, IPs_{AS_i}\}\})$$

Managing the blockchain infrastructure: A small monetary cost is defined for any protocol-specific related operation in the blockchain, i.e., *update* and *revoke* transactions that affect a certificate and not an AS identity (as these are for free). This is to force ASes to dedicate own computational resources to mine the blockchain and earn additional Certcoins needed to perform operations over the blockchain. Without miners the transactions could not be validated and the system would not function. A small cost per transactions prevents ASes from

submitting high volumes of requests that could result in miners, other ASes or *IANA*, not being able to detect and validate these illegal operations; even resulting in a DoS attacks. A rogue AS that tries to push into the blockchain a high amount of transactions will face two main impediments. First, it requires a significant amount of cryptocurrency to perform them which implies a high level of mining resources to earn that amount of cryptocurrency; second, these transactions will need between 7 and 10 minutes to be validated, which is the time required for miners in the blockchain to solve the problem that enables the validation of transactions, avoiding any potential DoS attack.

AS identity and credentials management: ASes use their online identity for every normal transaction over the blockchain. The offline identity is used if an AS wants to make changes over their own identity; for instance, revoke their online public key after a compromise is detected. Transactions over AS identities should be verified and subsequently signed by *IANA*.

Thus, *revoke* transactions without any protocol-specific information (such as the removal of a λ_i) and signed using an AS offline private key notify the rest of the ASes that the AS_i online public-private pair was compromised. IANA tracks this kind of transactions before its mining, signing them (to officially validate them) and providing the AS with a new SBTM-wallet disabling the previous wallet and revoking its associated certificate. Therefore, if AS_i wants to revoke its public key because it suspects that was compromised, it issues the following transaction: $(AS_i, revoke, online, P^{on}_{AS_i}, sign_{AS^{off}_i}\{AS_i, P^{on}_{AS_i}\})$. This transaction is verified using the offline public key of AS_i, $P^{on}_{AS_i}$, originally registered in the blockchain by *IANA*. Then, *IANA* verifies this revocation transaction. If it is valid, it will update the revocation transaction signing it to indicate that every AS that *IANA* has accepted the revocation and a new set of keys was issued for AS_i through a new SBTM-wallet:

$$(AS_i, revoke, online, P^{on}_{AS_i}, sign_{IANA}\{sign_{AS^{off}_i}\{AS_i, P^{on}_{AS_i}\}\})$$

At the point a new SBTM-wallet is given to AS_i, its offline secret key has to be also revoked by *IANA* through the following transaction:

$$(AS_i, revoke, offline, P^{off}_{AS_i}, sign_{IANA}\{AS_i, P^{off}_{AS_i}\}) \ .$$

After all these transactions are validated in the blockchain and a new SBTM-wallet is given to AS_i, AS_i generates brand new online $(P'^{on}_{AS_i}, S'^{on}_{AS_i})$ and offline $(P'^{off}_{AS_i}, S'^{off}_{AS_i})$ identities as it did when it received its initial SBTM-wallet. These identities will be communicated to *IANA* that registers them in the blockchain along with AS_i prefixes. Thus, the old compromised key gets revoked and a new identity is issued, allowing AS_i to operate securely again in the system.

ϕ_s operation over SBTM: With SBTM in place, ϕ_s can operate securely without any other trust management facilities.

soBGP. SBTM removes the need of distribution of soBGP *Certs*, as they are directly available in the blockchain. Whenever a border router needs to validate a routing message, it can directly retrieve the relevant information from

the blockchain and verify the message. This greatly reduces protocol message overhead compared to its standard implementation increasing its scalability, as there is no need for a permanent in-bound transmission of certificates. Furthermore, SBTM facilitates the validation of soBGP certificates and enhances their security; as they are stored and signed in the blockchain, certificates can not be modified or hijacked.

S-BGP. The blockchain replaces the two PKIs and the out-of-bound address attestation from S-BGP, thus reducing overhead and removing centralized trust management. For S-BGP over SBTM, when AS_i sends an UPDATE message to a neighbor AS_j which can verify the authenticity of the message by looking up AS_i's public key in the blockchain. Each AS in the path will add its signature in the route attestation. By validating these signatures, an AS can verify the path followed by the UPDATE message. Finally, with a quick lookup into the blockchain, a router can verify that the AS that advertised a certain prefix was, indeed, authorized to advertise it.

6 Brief Requirements Analysis

Requirement 1: SBTM maintains the security properties of the routing protocol, ϕ_s, it is deployed to support. SBTM does not change in any way the operation of ϕ_s as it is a complementary system to enable trust. It merely replaces PKIs and PGP systems.

Requirement 2: Every border router of an AS can retrieve the required security information. This is because security information is stored and managed in the blockchain. Every network device of an AS has an SBTM-wallet installed that provides access to this information.

Requirement 3: SBTM enables an incremental deployment from an insecure routing to its secure version. Once a secure protocol is to be deployed, SBTM flexibly allows seamless join of entities in the system, allowing ASes rollout ϕ_s at their own pace without penalizing other participants. Note that without an SBTM wallet, a router cannot validate ϕ_s messages from other ASes that already joined SBTM and configured their routers. Incoming messages can be validated once the wallet is installed at an AS_i, even though other transactions by or for AS_i are still pending.

Requirement 4: Adding new routers to the system is seamless and the blockchain allows every member of the system to access newly updated information in the blockchain once the transactions are validated. This also happens when newly updated information about router identities is added to the blockchain.

Requirement 5: Quick reaction to compromised ASes credentials is of utmost importance. SBTM is based on a blockchain protocol, benefiting from proof-of-work blockchain protocols intrinsic security properties. If any adversary tries to modify the information in any of the validated transactions of the blockchain, it will have to consequently modify the information of every previoulys validated transaction. All validated transactions in a proof-of-work

blockchain depend on its previous ones. This makes an attack over the blockchain infrastructure

An external adversary is not able to perform attacks over SBTM. It will not have access to the blockchain, removing its ability to forge information in the trust management system. An internal adversary comprise entities that are provided with an SBTM-wallet and therefore have access to the blockchain. These adversaries are recognized ASes of ϕ_s or adversaries that were able to hack and gain access to an AS border router and its installed SBTM-wallet[1]. Once a SBTM-wallet of a router R_i of AS_i has been hacked, all the identites of AS_i will be compromised as the SBTM-wallet enables access and control over all AS_i security information. Internal adversaries can try forging information in the blockchain. At this point, as all transactions are recorded, the compromised AS or other entities of the system will eventually detect that AS_i is misbehaving. This fact is signalled to IANA which is responsible for revoking and renewing AS_i SBTM-wallets and identities.

Every transaction is recorded in the blockchain allowing for misbehavior detection. An AS can periodically traverse its transactions in the blockchain to verify that no fake transactions were performed on its behalf violating its own policies. IANA also performs verifications periodically to ensure that, with all the information about ASes it has (IP ownerships, AS identities, etc.), no illegal transactions were performed. In addition, SBTM includes validation schemes and a fast way of renewing an SBTM-wallet and identities to minimize the harm of potential attacks over SBTM devices as already explained.

7 Performance analysis

To analyze the practicalty of SBTM we perform a preliminary performance analysis and discuss different dimensions of the framework.

Time per transaction: The time needed for a transaction to be verified and added to a block (through the mining process). Analyzing different types of proof-of-work based cryptocurrencies, the transaction validation time is of the same order of magnitude for all of them. If we consider the average validation time of Namecoin transactions, the cryptocurrency Certcoin is based on, the average validation time of a transaction is about 8.5 minutes per transaction. The time required for blockchain-based protocols to validate transactions change over time according to the status of the network. However, this time is around 10 minutes in a typical proof-of-work based blockchain such as Bitcoin. This does not suppose a problem for SBTM as the maximum validation time of a transaction and the amount of rewards miners get per validation can be initially configured in the genesis block of the blockchain during the SBTM blockchain deployment. However, a validation time of 10 minutes is useful to avoid potential

[1] SBTM-wallets represent a vulnerability of SBTM, as hacking this piece of software enables access to all the information stored in the blockchain and the victim AS_i information. Securing wallet software requires a seperate investigation.

attacks and the issuing of bursts of fake transactions, as already experienced in Bitcoin.

Evolution of the blockchain size: After a peak of initial transactions for the SBTM deployment, as a consequence of the registration of AS identities by *IANA* and the registration of protocol-specific information in the blockchain, it is difficult to infer the number of transactions per day that need to be performed in the system. Transactions after the initial stage will be revocation transactions, protocol-specific updates, the join of new routers for the already registered ASes or new ASes, as well as isolated identity registrations by *IANA* due to compromised keys or addresses. SBTM does not require posting any additional transactions for the operation of the system, as in other cryptocurrencies that are continuously performing transactions in the Blockchain such as Ethereum with its Smart Contract executions SBTM. The blockchain is used as a decentralized database that offers integrity and stores security information to enable trust. Taking into consideration the current number of ASes and prefixes, the size of the blockchain after the initial stage will be of approximately 50 MB. Then, taking the number of transactions per day performed in the already deployed decentralized DNS, Namecoin [1], approx. 300 transactions per day, the SBTM blockchain will grow by at most 300 KB every day. Such a blockchain size will not be a problem for modern routers.

Blockchain lookups: A challenge for SBTM is the time needed to perform lookups in the blockchain. The complexity of this operation will be of $O(n)$, where n is the size of the blockchain. However, the latest versions of Certcoin include certain improvements to reduce the lookup operations complexity. Specifically, a Distributed Hash Tables (DHT) scheme and cryptographic accumulators are used to reduce complexity. *Cryptographic accumulators* group user transactions to reduce the size of the blockchain from $O(s)$, in its basic version, to a $O(log\ s)$, where s represents the number of transactions in the blockchain. Thus, by reducing the size of the blockchain we reduce the time needed for lookup operations. On the other hand, the use of *Distributed Hash Tables* to manage the blockchain storage reduces the complexity of lookups from $O(n)$ to $O(log\ n)$. These two improvements, already included in Certcoin, significantly enhance the efficiency of lookup operations.

To have a practical approximation of the actual time needed to perform lookup operations over the blockchain, a set of experiments were conducted on a *2,5 GHz Intel Core i5* processor with *4 GB of 1600 MHz DDR3*. We check the time required to traverse every transaction in the blockchain for different blockchains sizes; this is a worst-case scenario of a lookup operation, with the transaction to be retrieved is at the end of the blockchain. Our results, in Table 2, show how, indeed, the complexity of a naïve lookup operation, without improvements grows linearly with the size of the blockchain. For sizes of representative blockchains for SBTM, without any scalability improvements, the time of a lookup operation is below 1 second. In practice, however, an AS will not have to traverse the entire blockchain. ASes typically make use of recently

updated information in the blockchain,"located" closer to the beginning of the blockchain. Moreover, certificate caches can reduce further lookup operations.

Table 2. Experimental full blockchain lookup time

Blockchain size	Full lookup time
350 MB	0.4 s
700 MB	0.83 s
1 GB	1.17 s
2 GB	2.33 s

SBTM computation cost: We analyze the UPDATE validation process in S-BGP when using SBTM as trust infrastructure. An analysis of S-BGP estimates that, on average, each UPDATE contains about 3.7 route attestations [12]. This requires, assuming that the router processing the UPDATE does not have any AS's address attestation and public key in cache: (a) one signature verification and one blockchain lookup to verify the authenticity of the UPDATE sent by a neighbor; (b) one blockchain lookup and one signature verification to obtain the address attestation (protocol-specific certificate) for each AS, and one signature verification to verify that the first AS in the route was allowed to advertise the prefix (in the UPDATE message); (c) one blockchain lookup and one signature verification for each RA to obtain each AS's public key and to obtain the prefixes they are allowed to advertise. Finally, another signature verification per route attestation to check their validity. On average, therefore, approximately 6 blockchain lookups and 13 signature verification needed per UPDATE message.

According to the traffic analysis in [6], UPDATE messages are received by a router ("BGP noise") at a rate between 50 and 200 per minute. This means that every SBTM-BGP router should be capable to validate at least 3 UPDATE messages per second. Each AS router will need to perform 39 signature verifications and 18 (3 times 6) lookups per second. Recalling the discussion from the previous point, we may infer that each blockchain lookup could performed in around 0.1 seconds, as the worst case scenario will require 1 second per lookup and all the useful information will be at the beginning of the blockchain. We conducted an experiment using the previously mentioned hardware to infer the maximum amount of hashes per second a device with these characteristics could perform: obtaining around 65.000 hashes per second. Routers can easily validate 3 UPDATE messages per second, thus holding a modest computational overhead sue to SBTM.

8 Conclusions

SBTM, a decentralized blockchain-based trust management system for inter-domain routing systems, can provide benefits in replacing currently considered

trust management systems for interdomain routing protocols without changes for secure routing protocols. An initial evaluation supports the practicality of SBTM, which facilitates gradual deployment of secure routing. Our on-going work considers practical constraints, further analysis, and expansion of the ideas presented here.

References

1. Namecoin blockchain information. https://bitinfocharts.com/namecoin/
2. Barrett, R., Haar, S.V., Whitestone, R.: Routing snafu causes internet outage. InterActive Week Online (Apr 1997), http://citeseerx.ist.psu.edu/showciting;jsessionid=2E4FE73B4D7FB05E73E53062377EC256?cid=651565
3. Blog, R.: Pakistan hikacks youtube, http://www.renesys.com/blog/2008/02/pakistan_hijacks_youtube_1.shtml
4. Boothe, P., Hiebert, J., Bush, R.: How prevalent is prefix hijacking on the internet. NANOG36 Talk, February (2006)
5. Butler, K., Farley, T., McDaniel, P., Rexford, J.: A survey of bgp security issues and solutions. Proceedings of the IEEE 98(1), 100–122 (2010)
6. Chuah, C.N., Bhattacharyya, S., Diot, C.: Measuring i-bgp updates and their impact on traffic. Tech. rep., Sprint ATL Technical Report TR02-ATL-051099 (2002)
7. Cohen, A., Gilad, Y., Herzberg, A., Schapira, M.: One hop for rpki, one giant leap for bgp security. In: Proceedings of the 14th ACM Workshop on Hot Topics in Networks. p. 10. ACM (2015)
8. Fromknecht, Velicanu, Y.: Decentralized public key infrastructure with identity retention. MIT (2015)
9. Goodell, G., Aiello, W., Griffin, T., Ioannidis, J., McDaniel, P.D., Rubin, A.D.: Working around bgp: An incremental approach to improving security and accuracy in interdomain routing. In: NDSS (2003)
10. Hollick, M., Nita-Rotaru, C., Papadimitratos, P., Perrig, A., Schmid, S.: Toward a taxonomy and attacker model for secure routing protocols. ACM SIGCOMM Computer Communication Review 47(1), 43–48 (2017)
11. Kent, S., Lynn, C., Mikkelson, J., Seo, K.: Secure border gateway protocol (s-bgp) real world performance and deployment issues,[in proc. isoc symp. network and distributed system security (ndss). San Diego, CA (2000)
12. Kent, S.T.: Securing the border gateway protocol. The Internet Protocol Journal 6(3), 2–14 (2003)
13. Papadimitratos, P., Haas, Z.J.: Securing the internet routing infrastructure. IEEE Communications Magazine 40(10), 60–68 (2002)
14. Rekhter, Y., Li, T., Hares, S.: A border gateway protocol 4 (bgp-4). RFC 4271 (2005)
15. Wählisch, M., Maennel, O., Schmidt, T.C.: Towards detecting bgp route hijacking using the rpki. ACM SIGCOMM Computer Communication Review 42(4), 103–104 (2012)
16. White, R.: Securing bgp through secure origin bgp (sobgp). Business Communications Review 33(5), 47–53 (2003)