Privacy Preserving and Resilient RPKI

Kris Shrishak
Technical University Darmstadt
Darmstadt, Germany

Haya Shulman
Fraunhofer SIT
Darmstadt, Germany

***Abstract*—Resource Public Key Infrastructure (RPKI) is vital to the security of inter-domain routing. However, RPKI enables Regional Internet Registries (RIRs) to unilaterally takedown IP prefixes - indeed, such attacks have been launched by nation-state adversaries. The threat of IP prefix takedowns is one of the factors hindering RPKI adoption.**

In this work, we propose the first distributed RPKI system, based on threshold signatures, that requires the coordination of a number of RIRs to make changes to RPKI objects; hence, preventing unilateral prefix takedown. We perform extensive evaluations using our implementation demonstrating the practicality of our solution. Furthermore, we show that our system is scalable and remains efficient even when RPKI is widely deployed.

I. Introduction

Resource Public Key Infrastructure (RPKI) [30] is a cryptographic method to secure inter-domain routing against prefix and sub-prefix hijacks. It is also a prerequisite for Border Gateway Protocol Security (BGPsec) [31]. In RPKI, Regional Internet Registries (RIRs) allocate IP prefixes and authorize specific autonomous systems (ASes) to be the origin of routes. This information is stored in route origin authorization (ROA). Routers use the ROAs to distinguish legitimate routes from leaked or hijacked routes. This is known as route origin validation (ROV).

The insecurity of inter-domain routing and the ability of RPKI to address the insecurity has not transpired into wide-scale deployment of RPKI [18], [24]. One of the reasons is the possibility of RIRs to unilaterally takedown IP prefixes, either deliberately or accidentally, that will result in the prefix of the affected ASes being unreachable when ROV is performed [11]. The hierarchical structure of RPKI gives RIRs the power to revoke and invalidate any objects that it has issued.

As centralized authorities are easy targets for legal surveillance and coercion, is it possible to prevent a state-sponsored attacker from imposing its demands on RIRs without drastically changing the structure of RPKI? The RIRs are bound by the law of the country they are based in. Their members who are based in different countries do not have a recourse when their prefix is taken down.

In the past, there have been situations where these problems have taken practical relevance. In 2011, RIPE NCC took the state of Netherlands to court when the Dutch police ordered to it to lock registration of four IP address blocks [38], [42]. Nevertheless, it was forced to lock down the registrations. More recently, RIPE NCC mistakenly deleted 2669 ROAs on 1 April 2020 and were reinstated on 2 April [39]. This meant that the announcement for these resources were 'unknown'. On the day when these ROAs were missing, RosTelecom had a route leak [35]. While the two events seem independent, according to RIPE NCC, 12 prefixes whose ROAs were deleted were affected by the route leak. Furthermore, RIPE NCC transferred an IP prefix block from a member to another entity based on a German court order transferred to them through a Dutch court [40]. As a matter of procedure, they will do the same if similar situations arises in the future. In the context of RPKI, this means RIPE NCC will "revoke any certificates generated by the RIPE NCC" [41].

In this work we address these issues that are prevalent in the deployed RPKI system by constructing a distributed RPKI system that relies on threshold signatures, a specific instance of secure multiparty computation (MPC). Our solution, without requiring significant changes to BGP and RPKI, restricts the power of RIRs and only allows revocation of allocated resources in legitimate cases with the cooperation of a number of RIRs.

A. Significance of the threat model

BGP without RPKI operates in a default-accept mode where any autonomous system (AS) can announce a BGP route for any IP prefix and the other ASes will accept the route by default. The default-accept mode has made BGP vulnerable to prefix hijacks, where a malicious AS announces a route for IP prefixes it does not own such that the traffic for those prefixes are sent to it, and sub-prefix hijacks, where a malicious AS announces a more specific IP prefix than the one that has been allocated [2], [7], [13], [26].

RPKI entrusts hierarchical and centralized authorities to be honest. Malfunctions or coercion by law enforcement authorities is not incorporated into the threat model. Such a weak threat model creates an imbalance of power between the RIRs and its members. Moreover, the power imbalance with RPKI is greater than with Web PKI. In RPKI, there is no option to request certificates from different authorities. Hence, the reliance on specific RIRs is greater.

Members are further weakened when the authority is based in a different country than their own. The manipulations at the level of BGP is more coarse-grained than domain name seizures as BGP granularity is limited to $/24$, i.e., 256 IPv4 addresses [11]. The RIRs are bound by the law of the country they are based in. If members are affected, they may need to take the issue up in another country. The slow process may result in a loss of business.

B. Threshold signatures for RPKI

We propose a distributed RPKI system based on threshold signatures. Threshold signatures is a cryptographic technique where a threshold of $t + 1$ parties out of a set of n parties are required to jointly compute a signature on a message. Although the signing process is distributed, the verification scheme remains unchanged. Threshold signatures are more robust to adversarial attacks in settings where signatures are to be generated in a system where individual parties cannot be entirely trusted.

Threshold signatures provide a method to distribute trust and they are practical in settings where the number of participating parties is small. One of the deployment method of RPKI is *hosted* RPKI. There are five RIRs and threshold signature protocols are practical when there are only five participating parties. Hence, our system requires a threshold of them to agree before making changes to RPKI objects. This prevents any RIR from unilaterally making changes.

Our solution can be described as follows: threshold signatures use shares of the private key, where each of the five RIRs will have a share of the private key while none of them have the entire private key. Using only the shares, the RIRs can collaboratively sign ROAs and CRLs. However, they cannot unilaterally perform any of these actions. Our mechanism prevents them from acting maliciously unilaterally. Most importantly, threshold signatures support a stronger threat model where corrupted RPKI authorities are not entirely trusted and yet play a significant role in making BGP secure.

Contributions. A summary of our contributions:

- We construct a distributed RPKI system based on threshold signatures that addresses three issues: (1) preventing unilateral IP prefix takedowns, (2) limiting the scope and implications of attacks on RIRs, and (3) enabling validation in case of missing trust anchor.
- We propose two deployment models of our solution and discuss the trade-offs in these models.
- We show the performance of our distributed RPKI system based on four threshold signature protocols, all of which have a stronger threat model than the existing RPKI system.
- We perform extensive evaluation of our system and show that our system is not only efficient for today's requirements, it can also meet future demands.

Outline. We provide preliminaries in Section II. We elaborate on the system and threat model and describe our distributed RPKI system in Section III. Then we discuss the performance of our distributed RPKI system in Section IV. In Section V, we analyse historical RPKI data to understand the number of ROAs issued/revoked over time and show that our system satisfies the requirements. Finally, we discuss the related works in Section VI and we conclude in Section VII.

II. Preliminaries

A. RPKI

RPKI architecture includes CA certificates, end-entity (EE) certificates and trust anchor. A resource holder needs a CA certificate to sub-allocate resources and to issue resource certificates. EE certificates verify signed objects (e.g., ROAs and manifests). The private key corresponding to the public key in an EE certificate cannot be used to sign other certificates. There is a one-to-one mapping between EE certificate and signed objects. If the EE certificate is revoked, then the corresponding signed object is automatically revoked. CA certificate is used to sign EE certificate. A trust anchor is a self-signed X.509 CA certificate in RPKI that is at the head of the chain and it is assumed to be trusted. In X. 509 architecture, the chain of trust is derived from this authoritative certificate. The trust anchor contains a public key in the `subjectPublicKeyInfo` field along with the associated data that are used by the relying parties to validate a signature on a certificate or signed objects, such as ROAs [27], [36]

ROAs are digitally signed objects, X.509 certificates [32], [12], that provide a method to verify that an IP address block holder (RIR) has authorized an AS to originate routes to specific prefixes within that address block. Note that each ROA includes exactly one ASN. However, multiple ASNs may be authorized, but each one requires a separate ROA. Moreover, issuance of subordinate certificates corresponds to sub-allocation of IP-addresses. A Certificate Revocation List (CRL) is a list of resource certificates that have been revoked, and should not be relied upon by the relying parties. A CRL is always issued by the same CA that issues the corresponding certificates.

There are two RPKI models: delegated RPKI and hosted RPKI. In the delegated RPKI model, AS runs a CA as a child of RIR (or NIR or LIR), generates its own certificate, gets it signed by the parent CA. This model allows the AS to operate independent of the parent RIR. For large operators of a global network, this model is suitable so that they do not need to maintain ROAs through the different web interfaces of the RIRs. However, this model is not suitable for all as it requires running a CA and maintaining the ROAs.

In the hosted-RPKI model, RIRs host the CA, that is, the same entity that allocates IP resources also runs the CA to validate the ROAs. Thus, in this model, they are trust anchors. In a way, this is meaningful as the RIRs already know the owner of the address space. Existing RPKI systems are tied-up with the login credentials of the ASes at the RIR. Signing and key rollover is automatic. It is easy for the owners of the address space to begin using hosted RPKI than delegated RPKI as the CA functionality is taken care of by the RIR. This model is convenient for most ASes. It is easier to use and it is especially useful for members with a small network and with limited resources. Even large providers such as Cloudflare make use of hosted RPKI [1]. Furthermore, the RIR assumes responsibility to publish the signed objects. However, this convenience comes at the cost of further centralization of power as the RIRs also handle the private keys used to sign ROAs.

[1] https://ripe77.ripe.net/presentations/156-RPKI-deployment-at-scale-RIPE-1.pdf

B. ECDSA

The scheme is parameterized by a curve point G of prime order p, and we write $\mathbb{Z}_p$ for the field of order p. We use H to denote a function mapping arbitrary length messages unto elements of $\mathbb{Z}_p$.

Key Generation. $\mathsf{KGen}(1^\lambda)$

1) Sample at random $\mathsf{sk} \leftarrow \mathbb{Z}_p$ as the signing key.
2) Compute $\mathsf{pk} = \mathsf{sk} \cdot G$ as the public verification key.
3) Output $(\mathsf{sk}, \mathsf{pk})$.

Signing. $\mathsf{Sig}(\mathsf{sk}, M)$

1) Sample at random an instance key $k \leftarrow \mathbb{Z}_p$.
2) Compute $R = (r_x, r_y) = k \cdot G$. If $r_x \equiv 0 \pmod p$, go back to step 1.
3) Compute $s = k^{-1}(H(M) + \mathsf{sk} \cdot r_x)$ where H is a hash function.
4) Output $\sigma = (r_x, s)$.

Verification. $\mathsf{Vf}(\mathsf{pk}, M, \sigma)$

1) Let $(r'_x, r'_y) = s^{-1}(H(M) \cdot G + r_x \cdot \mathsf{pk})$.
2) Output $r'_x = r_x$.

Correctness. We have

$$
\begin{aligned}
& s^{-1}(H(M) \cdot G + r_x \cdot \mathsf{pk}) && (1)\\
&= k(H(M) + \mathsf{sk} \cdot r_x)^{-1}(H(M) \cdot G + r_x \cdot \mathsf{pk}) && (2)\\
&= k \cdot G \cdot ((H(M) + \mathsf{sk} \cdot r_x)^{-1}(H(M) + \mathsf{sk} \cdot r_x)) && (3)\\
&= k \cdot G = (r_x, r_y), && (4)
\end{aligned}
$$

which shows that valid signatures verify.

C. MPC and threshold signatures

In our work, we use threshold signature protocols that are based on secret sharing. Specifically, we use additive sharing and Shamir sharing schemes. We use the notation $[a]$ to denote a value a that is secret-shared, that is, no single party can access it. For $a \in \mathbb{Z}_p$, the shares $[a]$ are also elements of $\mathbb{Z}_p$. We use the command Open to reconstruct from the secret shared values such that $a \leftarrow \mathsf{Open}([a])$ For malicious security, we use message authentication code (MAC) scheme of SPDZ [15], [16].

In SPDZ, a value a is represented as $[a] = ((a_1, \ldots, a_N), (\gamma(a)_i, \ldots, \gamma(a)_N))$ where a_i is a share of a and $\gamma(a)_i$ is the MAC share authenticating a under a global key $\alpha \in \mathbb{Z}_p$ such that $a = \sum_i a_i$ and $\alpha \cdot a = \gamma(a) = \sum_i \gamma(a)_i$. Each party i holds the pair $(a_i, \gamma(a)_i)$. The execution of Open in SPDZ involves the broadcast of the shares a_i by each party and computing $\sum_i a_i$. Then the MAC is checked to confirm that a is correct. For this check, each party computes $\gamma_i(a) - \alpha_i a$, broadcasts the commitment and checks if $\sum_i = \gamma_i(a) - \alpha_i a$.

Secure computing of ECDSA signatures does not only require the secret key sk to remain secret from all the parties but also the instance key k. The computation of k^{-1} should also be performed securely so that information about k is not revealed. This is also the the most computationally expensive part of securely computing ECDSA signatures.

III. Distributed RPKI

A. Threat Model

In our distributed RPKI system, we consider a stronger threat model than the existing RPKI system. The existing threat model of RPKI includes external adversaries, but not the participating entities, such as RIRs, to be a possible attacker. In this work, in addition to the threats considered in the existing system, we do not consider the RIRs to be entirely trustworthy.

Our threat model accounts for mistakes by the RIR as the hosted CA, the RIR under attack from an external adversary including legal coercion to modify, revoke or to inject RPKI data. All these scenarios require access to the signing key for the attack to work. We can capture these scenarios in our system by incorporating RIRs in the threat model. Note that attacks on the publication point, such as deletion of RPKI data, are beyond the scope of this work.

Standard MPC terminology provides us with a tool kit to discuss threat models that not only includes external adversaries but also the participating parties. Thus, we introduce standard MPC terminology to describe threat models in a distributed setting. We consider adversary power, that is, whether an adversary is passive or active. Then, we describe the guarantees that can be achieved when the threshold of honest parties varies. Finally, we describe which guarantees our solution supports and how it translates to the threats against RPKI.

a) Honest-but-curious vs. malicious security.: MPC protocols can be classified in terms of the power of the adversary. An adversary can be *honest-but-curious* or *malicious*. An honest-but-curious adversary follows the protocol while a malicious adversary does not follow the protocol and might actively disrupt the protocol. Security against honest-but-curious adversary is sufficient in many real life scenarios. If the RIRs trust each other not to act maliciously and instead consider each other to be a necessary check on each other's operation, a protocol secure against honest-but-curious adversary is sufficient. Such a protocol keeps the signing key away from any internal adversaries and curious employees at the RIRs. On the one hand, a malicious adversary has full control over the RIRs and can disrupt the protocol. For instance, it can send wrong values or delay the sending of values. On the other hand, honest-but-curious adversaries are assumed to provide the correct inputs during the protocol so that no checks on the correctness of the inputs needs to be performed. As might be obvious, a protocol against malicious adversaries provides stronger security guarantees.

b) Honest vs. dishonest majority.: Let n be the number of RIRs participating in the distributed RPKI system. During the execution of a threshold signature protocol, a threshold t number of RIRs need to be available for the protocol to be successfully executed. When a majority of the RIRs are honest, a threshold $t \leq \lfloor (n-1)/2 \rfloor$ of parties are needed to sign, then it is called *honest majority*. When a minority of the RIRs are honest, that is, $t < n$, the protocol is said to be secure for a *dishonest majority*. In the case of honest

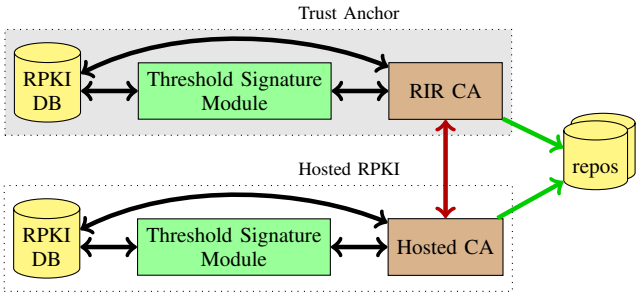

Fig. 1. System setup

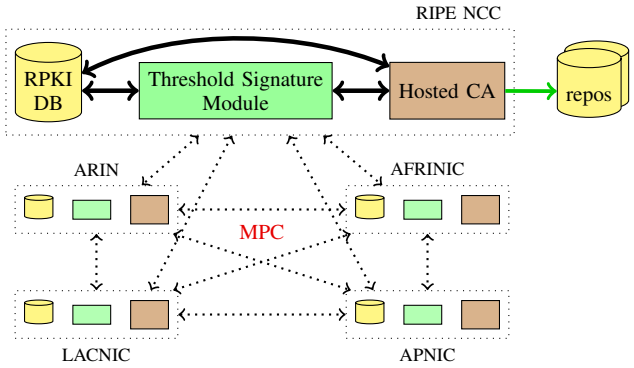

Fig. 2. Distributed RPKI architecture

majority protocols, as long as three-out-of-five RIRs do not collude, the key is not disclosed to anyone. As the RIRs often do not converge on the same policies, this may not be a strong assumption [33]. However, there are situations where a dishonest majority protocol might be needed as it provides stronger security such that the adversary needs to corrupt all the parties to be able to access the signing key. A dishonest majority protocol might also be required when the signature should only be created when there is unanimity among the RIRs.

B. System setup

We present the system architecture in this section. The setup at each RIR is shown in Figure 1 and the distributed RPKI architecture is shown in Figure 2. Each RIR has two components: Trust anchor and Hosted RPKI. Each of them has a CA, a threshold signature module and access to a local RPKI database. All the certificates and the signed objects issued by the two CAs are published in public access repositories, through rsync or RRDP [6].

Our system incorporates all parts of the RPKI that requires generating signatures, which includes the creation of signed objects, ROAs, as well as signing of the resource certificates of children and the issuance of CRLs. We distribute the task of creating signed objects among multiple RIRs. As there are five RIRs, we use $n = 5$ in our system.

We focus on the key generation and the signing operation in a distributed RPKI system, such that no RIR has access to the signing key. RIRs create their shares of the secret key and have access only to these share and not to the secret key. For communication between the RIRs, we assume the existence of

Key generation $\mathsf{KGen}(1^\lambda)$

1) Each RIR takes a security parameter 1^λ as the input and generates a signing key share for the j^{th} member by randomly sampling $[\mathsf{sk}_j] \leftarrow \mathbb{Z}_p$.
2) Each RIR locally converts $[\mathsf{sk}_j]$ to $[sk_j] \cdot G$.
3) RIRs compute the public key $\mathsf{pk}_j = \mathsf{Open}([sk_j] \cdot G)) = \mathsf{sk}_j \cdot G$.
4) Output the secret key shares and the public key $([\mathsf{sk}_j], \mathsf{pk}_j)$.

Fig. 3. Key generation protocol

a synchronous communication network and that the protocols are run on a point-to-point network [9], [28]. We also assume that the RIRs use a secure and authenticated communication channel, e.g. using TLS.

C. DRPKI protocol phases

In our system, each RIR has a share of a private key for each member, uses this share to collaboratively issue signed objects and does not have access to the entire private key. The protocols we use in this paper are in the security-with-abort model. In this model, the protocol aborts if the participants are not available. We note that threshold signature protocols that give guarantees on availability are possible.

The interactive protocols in our system are run between the All phases instigated by a Hosted CA and the interaction takes place between the Threshold Signature modules. Due to the computation and communication overhead, we need an efficient threshold signing protocol. This means, that when the signature is to be generated, there needs to be as little overhead as possible in comparison to traditional signatures. This is possible with protocols that move most of the cryptography to the preprocessing phase and requiring minimum processing in the online phase, when the message to be signed is available. Furthermore, we consider the efficiency in light of the threat models we discussed in Section III-A. Based on these requirements, we adapt the threshold signature protocol of Dalskov et. al. [14] for our purpose (Figures 3–6).

1) Key generation: In the key generation phase, new keys are generated such that each RIR generates a signing key share $[\mathsf{sk}_j]$ for each member and runs the key generation protocol. At the end of this phase, each RIR has the public key pk_j and their share of the signing key $[\mathsf{sk}_j]$. The key generation protocol needs to be run every time keys are to be generated. The keys do not need to be stored in a HSM. The complete signing key is not exposed unless a threshold number (depending on the protocol being *honest majority* or *dishonest majority*) of RIRs have been compromised. Figure 3 describes the protocol details.

2) Signing: The threshold signing protocol we use has two preprocessing phases and one online phase. The first preprocessing phase is independent of the member for whom

Member independent preprocessing

For each signature to be generated,

1) RIRs generate tuples of secret shared values of the form $([a], [b], [c])$ such that $a, b, c \in \mathbb{Z}_p$ where $c = ab$.
2) They open the share $[c]$ by running $c \leftarrow \mathsf{Open}([c])$.
3) Let $[k^{-1}] = [a]$.
4) Each RIR locally generates $\langle k \rangle = ([b] \cdot G) \cdot c^{-1}$.
5) Output the initial preprocessing tuple $(\langle k \rangle, [k^{-1}])$.

Fig. 4. Member Independent preprocessing

Member dependent preprocessing

1) RIRs input the generated signing key shares $[\mathsf{sk}_j]$ and the initial preprocessing tuple $(\langle k \rangle, [k^{-1}])$.
2) They compute $[\mathsf{sk}'_j] = [\mathsf{sk}_j/k]$ by generating an additional tuple and Beaver's rerandomization technique [3].
3) Output the final preprocessing tuple $(\langle k \rangle, [k^{-1}], [\mathsf{sk}'_j])$.

Fig. 5. Member Dependent preprocessing

the signature is to be generated. More specifically, this phase is independent of the signing key to be used. This property allows us to amortize this phase. This phase can be run between the RIRs before the member's request to generate a signature arrives. Only an estimation of the number of signatures that would be required in a certain amount of time is required to run this phase. At the end of this phase, the desired number of initial preprocessing tuples are generated and stored at each RIR. Figure 4 describes the protocol details.

The second preprocessing phase is dependent on the member for whom the signature is to be generated. The threshold signature modules at the RIRs use the one unused initial preprocessing tuple. It is security critical that the initial preprocessing tuples are not reused as it is equivalent to the reuse of the instance key k. At the end of this phase, the desired number of final preprocessing tuples are generated and stored at each RIR. Figure 5 describes the protocol details.

In the final signing phase, the member gives consent to the changes that can be made through a standalone application. This consent is sent to all the RIRs. When a signature is to be generated, the message to be signed is sent by the RIR that initiates the signing protocol to the other RIRs. The message is checked, similar to the checks each RIR performs in the existing RPKI system, where they check the message locally for the message before they individually sign. However, in our case, the check is performed by all the RIRs for all the messages that need to be signed. Furthermore, the consent of the member is checked. The RIRs check whether

Final signing phase

1) The member uses a standalone application to give consent, e.g., to transfer IP-space to another AS. The consent is sent to all the RIRs.
2) Input the message to be signed M and the final preprocessed tuple $(\langle k \rangle, [k^{-1}], [\mathsf{sk}'_j])$.
3) The RIR initiating the protocol sends the message M to the other RIRs.
4) The RIRs check the contents of M and the consent by the member before proceeding. If the check fails, they abort $\perp$. Else, they continue.
5) Then the RIRs compute $R \leftarrow \mathsf{Open}(\langle k \rangle) = (bc^{-1}) \cdot G = a^{-1} \cdot G = k \cdot G$.
6) Let $(r_x, r_y) \leftarrow R$.
7) Locally compute the share of the signature $[s] = H(M) \cdot [k^{-1}] + r_x \cdot [\mathsf{sk}'_j]$.
8) Finally compute the signature $s \leftarrow \mathsf{Open}([s])$ and output $\sigma = (r_x, s)$ or $\sigma = \perp$.

Fig. 6. Final signing phase

the consent has been given for the specific change, e.g., the transfer of IP space to another AS. Note that a transfer of IP-space requires consent for a CRL for the existing EE certificate associated with the ROA and to create a new signed ROA. These checks prevent RIRs to unilaterally take decisions to revoke certificates. If a threshold number (depending on the protocol being *honest majority* or *dishonest majority*) of RIRs agree, then the RIRs locally compute their share of the signature before jointly computing the final signature. Figure 6 describes the protocol details. With regards to the format of the messages, we do not make any change to the form and fields compared to the existing RPKI system. The certificates take the form of X. 509 certificates [25] while the signed objects conform to RFC 6788 [29]. The RIRs check the contents of the message out-of-band.

3) Automation: Our system is fully automated and does not need manual intervention in its regular operation. Step 4 in Figure 6 checks the message before it is signed. This check, performed by all RIRs, verifies whether a consent has been received from the INR holder in Step 1. If a threshold number of RIRs have not received the consent, then the check fails and the automated signing protocol aborts.

4) Legitimate revocation without consent?: So far, we have assumed that revocation of allocated IP resources requires the consent of the INR holder. What about cases where there is a legitimate reason to revoke allocation? Let us take a case where ARIN was fraudulently induced to issue IPv4 addresses [1]. After the fraud was detected and ARIN won a legal case, ARIN was able to take back the addresses. Using our automated system with enforced consent, revocation of the IP address space in such a scenario will not be possible. However, we are able to accommodate legitimate revocation with a minor change to the system.

Our automated system aborts the protocol if the check for consent fails at Step 4 in Figure 6. Instead of aborting the protocol, we can flag it with the requirement for manual intervention if the protocol is to be completed. Note that such a manual intervention will not require a large human effort as, in practice, most organizations obtain IP address space from their RIRs in good faith and there are only a few bad apples [43]. We will require the RIRs to communicate off band before the protocol is completed. This mechanism also allows for legitimate law enforcement requests to be processed by the RIRs, only when a threshold of other RIRs also agree. Note that although technically possible, processing law enforcement mechanisms in this manner is akin to private regulation, which will require legal and policy changes for it to be realistic.

D. Deployment scenarios

We propose two different deployment models. We begin with a naïve deployment model and explain the reasons for its failure to solve the problem. Then, we present two solutions with their associated trade-offs among the stake holders. We emphasise that the trade-offs are not with respect to the security, but with respect to the responsibilities of the different stake holders.

Naïve solution: In a naïve solution, our threshold signing module can be used for Hosted CAs. This solution allows for the existence of the delegated CAs, which are beneficial to ISPs who sub-allocate resources. This solution allows for IP-space owners to run their own CAs as well, that is, delegated CA system can continue to function in parallel with hosted CA system, as it does in the current system. So the ISPs which have their own CA can delegate IP-space and sign the ROAs. However, only change is that the signing keys in the hosted CA setup are not in the possession of the individual RIRs. The trust anchor from the existing RPKI exists and the RIR CA which is higher in the hierarchy can still revoke certificates unilaterally as it is not distributed. And, the threat model remains weak and unchanged.

Our solutions: As the naïve deployment scenario does not solve the problem, we propose two deployment solutions. Our first solution prevents unilateral takedown by the RIRs while our second solution also prevents LIRs from unilaterally taking down prefixes. Both our solutions distribute the trust anchor. Before discussing our solutions, we give an intuition behind our choice to distribute RPKI trust anchor. The notion of a trust anchor requires all child nodes to unconditionally trust an entity. In RPKI, there are five trust anchors, one at each RIR, which the relying parties use to verify RPKI signatures. The concentration of power at trust anchors in the internet infrastructure extends beyond RPKI and is also observable in DNS(SEC) and Web PKI. Although many of the problems and vulnerabilities are similar [4], [5], [45], [46], unlike DNS and Web PKI, there are already five trust anchors in RPKI that allows for a smooth transition to a distributed trust anchor. Furthermore, the existing system of five trust anchors has had its issues. As the policies of each RIR with regards to trust anchor is different, some relying parties do not use the trust anchor of ARIN and ROAs issued under ARIN's trust anchor locator fall to the status of 'Not Found' [47]. This means that even when RPKI is implemented, a significant portion of the networks do not validate routes originating from North America due to policy decisions and legal barriers [48]. Thus, in practise large parts of the world are prevented from having better routing security. These issues can be prevented if the trust anchor is not located at individual RIRs with their own policies and is instead distributed across them.

Hierarchical deployment: In our first solution, we propose a two-layered tree deployment that maintains the hierarchical structure of RPKI. In both layers, the RIRs use our threshold signature module. The upper layer generates a distributed trust anchor to the five RIRs, while the lower layer uses the threshold signing module for the Hosted CAs. In the upper layer, a distributed trust anchor is established using our key generation protocol in Figure 3. Each RIR generates their signing key share and participates in the key generation protocol to obtain the public key. Once the public key is obtained, each RIR adds the public key to their TAL as the `subjectPublicKeyInfo` [12]. Each RIR has a TA that has the same public key in the TAL. As no RIR has the private key associated with this certificate, the RIR CAs do not need to be kept offline. Thus, the RIRs do not need a subordinate CA to issue child certificates. Furthermore, as each RIR has the same public key as part of the trust anchor and they have the same `subjectPublicKeyInfo` in their TAL, access to the TAL from one RIR is sufficient for relying parties to validate routes originating from any part of the world that has deployed RPKI. Note that to generate the child certificates, RIRs run the signature generation protocol described in Section III-C2.

In the lower layer, our threshold signing module is used by the Hosted CAs to generate signed objects such as ROAs. We are able to support delegated CAs as the distributed trust anchor at the RIR CAs is used to generate child certificates. Furthermore, this solution allows for incremental deployment as the LIRs who have already deployed their own CAs can continue to use them to serve their child nodes while those who have not deployed their own CAs can start using hosted CA. Note that the concerns regarding some LIRs being coerced by their country of registration remains.

Flat deployment: In our second solution, instead of having the RIRs run two CAs, RIR CA and the Hosted CA, we combine the two so that the RIRs only need to run one CA. Furthermore, we do not need a trust anchor as we replace the top-down architecture with a flat deployment architecture. Not only do we eliminate the hierarchical structure of existing RPKI, we also distribute trust. Moreover, this solution accounts for a stronger threat model where none of the CAs need to be completely trusted. However, we do not support delegated CAs in this solution. The CAs only generate end-entity certificates and signed objects; they do not generate any CA certificate that will allow child nodes to generate their own signed objects. This also means that child nodes will need the RIRs to generate signed objects for their child nodes. Nevertheless, we prevent any single entity to be all powerful

Majority \ Adversary power	Honest-but-curious	Malicious
Honest	Shamir	Mal. Shamir
Dishonest	Semi. OT	MASCOT

TABLE I
FOUR MPC PROTOCOLS

and require the participation of a threshold number of RIRs for a signed object to be generated and ejected.

IV. Implementation and Evaluation

We have implemented our system in C++ and have used MP-SPDZ [17] for the threshold ECDSA MPC protocols. MP-SPDZ includes threshold ECDSA protocol implementations for all the security models that we are concerned with: honest-but-curious and malicious as well as honest and dishonest majority protocols. In particular, we use four protocols—Shamir, Mal. Shamir, Semi OT and MASCOT—that are shown in Table I. The former two are based on Shamir secret sharing while the latter two are based on additive secret sharing. We use all four protocols to implement the system described in Section III-C.

A. Deployment Setups

For performance evaluation, two deployments were set up. For each node, we used an Amazon AWS c5.2xlarge instance with a 64-bit Intel Xeon CPU with 3 GHz and 16 GB RAM. We run all the evaluations on a single thread. To make our evaluations as realistic as possible, we chose to run the experiments based on the location of the RIRs. The five RIRs are in different continents of the world. So, in the first setting, we run experiments on five Amazon AWS instances that are placed around the world such that they are representative of the location of the RIRs. Specifically, we use the instances at Frankfurt, N.Virginia, Sydney, Sao Paolo and Mumbai while the RIRs are based in Amsterdam, Virginia, Brisbane, Sao Paolo, Mauritius, respectively. The latency and the bandwidth between the instances is shown in Figure 7. Furthermore, we also consider the setting where the RIRs could, in the future, have virtual servers located close to other RIRs. For this purpose, we also run our experiments on the LAN in Frankfurt.

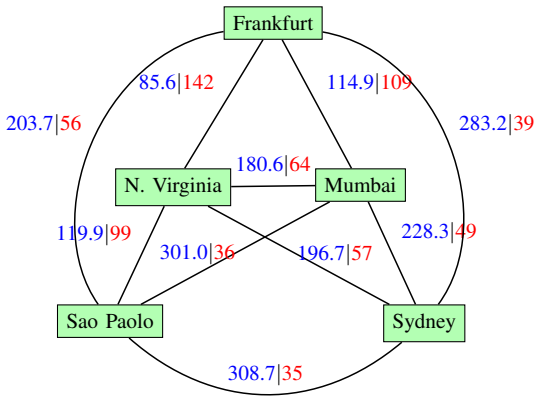

Fig. 7. Latency|Bandwidth between regions, where latency is in milliseconds and bandwidth is in Mbits/s

B. Experimental evaluations

Key generation. We benchmark the 5-party key generation protocol in both the settings. The total key generation time is composed of the timings for generating secret key and public key. Secret key generation involves generating a field element $[sk]$ while public key generation involves a local conversion of the field element into an elliptic curve point of order before being opened. The timings shown in Table II are the mean and standard deviation over 10 executions of the protocols where the value taken for each execution is the time noted when the last party completes the protocol. While the honest majority protocols (Shamir and Mal. Shamir) only require one round of communication for secret key generation, dishonest majority protocols (Semi OT and MASCOT) are costlier, especially in WAN setting.

Signing. We benchmark the preprocessing time (member dependent and independent) to generates tuples and the online signing time per signature in Table III. For preprocessing, we present the time taken to generate one tuple when 1000 tuples are generated in an amortized manner. As the preprocessing does not depend on the message to be signed, thousands of preprocessed tuples can be generated and stored. They can be used when a new message is to be signed. Note that the online phase does not involve any elliptic curve operation and, hence, is computationally cheap.

Although dishonest majority protocols are generally costlier than honest majority protocols, Semi OT has the highest preprocessing throughput in LAN setting (Table IV). Semi OT protocol uses additive sharing which is cheaper than elliptic curve operations, which is the predominant cost during preprocessing. In the WAN setting communication becomes more predominant than local operations. We also observe that the cost of malicious security in the case of honest majority protocol is very small. This is especially true in the WAN setting as the extra checks for Mal. Shamir are local operations while communication becomes the predominant cost.

In Table V, we show the communication per party for the four protocols. We note that the communication is asymmetric for Mal. Shamir and Shamir. Hence, we present the mean of the communication over all the parties. We notice that the preprocessing communication per tuple as well as online signing is significantly higher for dishonest majority protocols than honest majority protocols. In comparison, the communication overhead per party is marginal for malicious security over honest-but-curious protocols.

V. Analysis

For the deployment of our distributed RPKI system, it needs to be efficient enough. In the previous section we discussed the efficiency in terms of the runtime of our protocols. In this section, we discuss whether they are efficient enough in terms of the number of signatures required by the RIRs. As our system involves all the five RIRs, we take into account the cumulative of the requirements of all of them.

a) RPKI data: We accessed the publicly available historical RPKI data maintained by RIPE NCC that includes the

	LAN			WAN		
	Secret	Public	KGen	Secret	Public	KGen
MASCOT	6.99 ± 0.04	2.48 ± 0.02	9.47 ± 0.03	4490 ± 1.74	1147 ± 0.27	5637 ± 1.25
Semi OT	0.88 ± 0.04	0.91 ± 0.02	1.79 ± 0.03	851 ± 2.53	486 ± 0.93	1337 ± 1.91
Mal. Shamir	0.24 ± 0.00	1.59 ± 0.03	1.83 ± 0.02	198 ± 0.23	487 ± 0.61	685 ± 0.46
Shamir	0.25 ± 0.04	1.13 ± 0.08	1.38 ± 0.06	284 ± 1.38	382 ± 3.48	666 ± 2.64

TABLE II
BREAKDOWN OF KEY GENERATION TIMINGS IN MILLISECONDS FOR [sk] SHARING, AND pk.

	LAN			WAN		
	Preprocessing	Online	Sig	Preprocessing	Online	Sig
MASCOT	4.78 ± 0.01	1.89 ± 0.02	6.67 ± 0.02	50.56 ± 1.86	1055 ± 37.23	1106 ± 26.36
Semi OT	0.96 ± 0.01	1.51 ± 0.01	2.47 ± 0.01	9.00 ± 0.90	487 ± 0.40	496 ± 0.70
Mal. Shamir	1.43 ± 0.00	1.40 ± 0.02	2.83 ± 0.01	10.94 ± 0.68	283 ± 0.06	294 ± 0.48
Shamir	0.98 ± 0.00	1.30 ± 0.02	2.28 ± 0.01	3.77 ± 0.00	282 ± 0.18	286 ± 0.13

TABLE III
BREAKDOWN OF SIGNING TIMINGS IN MILLISECONDS FOR PREPROCESSING AND ONLINE PHASES PER SIGNATURE. PREPROCESSING TIMES ARE BASED ON AMORTIZED GENERATION OF 1000 TUPLES.

	LAN		WAN	
	Preprocessing	Online	Preprocessing	Online
MASCOT	209	529	20	0.95
Semi OT	1042	662	111	2.05
Mal. Shamir	699	714	91	3.53
Shamir	1020	769	265	3.54

TABLE IV
BREAKDOWN OF THROUGHPUT FOR PREPROCESSING (TUPLES/SEC) AND ONLINE PHASES (SIGNATURES/SEC).

	KGen	Preprocessing (per tuple)	Online Signing
MASCOT	0.482	624	0.400
Semi OT	0.113	99.0	0.128
Mal. Shamir	0.271	1.345	0.0768
Shamir	0.206	0.437	0.0512

TABLE V
COMMUNICATION PER PARTY (KBYTE)

daily archive of the repositories of all the five RIRs from 2011 onwards [2]. We use the historical data from 11 March 2015 till 10 August 2020.

b) ROA analysis: We use the RPKI data to analyse the number of ROAs that have been added and removed per day in a certain time period. We estimate the number of signatures required based on this information. Figure 8 shows the change on average day (mean taken over a month) in ROAs for the five RIRs. On average, we need about 8000 signatures per day. However, there are days when the load is greater. This occurs on days when many ROAs are re-issued. Figure 9 shows the maximum number of changes per month. Note the scale on y-axis: There is a twenty times difference from Figure 8.

We observe from Table IV that for our slowest protocol MASCOT, we are able to produce 0.95 signatures/sec or 82080 signatures/day in the WAN setting. For our fastest protocol, we are able to produce 3.54 signatures/sec or 305856 signatures/day in the WAN setting. Even our slowest protocol can produce 10x more signatures than is required on an average day. All our other protocols are fast enough even on days with peaks in Figure 9. In the LAN setting, all our protocols are fast and have the capacity to produce three orders of magnitude more signatures. The efficiency of our system makes it possible to scale it as the adoption of RPKI increases.

VI. RELATED WORK

Threshold Signatures for Internet Infrastructure. Threshold signatures for DNSSEC have been considered in the past. Cachin and Sanar [8] proposed a distributed DNS to avoid single point of failure, while Cifuentes et. al [10] use threshold signatures to emulate a HSM at an authoritative name server. Dalskov et. al [14] use threshold signatures to secure DNSSEC signing keys when the zone and key management is outsourced to DNS operators. Integration of new algorithms into DNSSEC can be done with ciphersuite negotiation mechanisms and similar ideas can be applied for RPKI [23], [21], [22]. Finally, Shrishak and Shulman [44] initiated the research direction of using threshold signatures for RPKI. We extend that work further by developing and setting up a complete system in the Internet along with extensive performance evaluations.

Limiting the power of RIRs. There have been two approaches in the prior works to limit the power of RIRs in RPKI. One approach has been to add transparency logs and `.dead` objects to RPKI to note the consent of the Internet Number Resource (INR) owner for revocation [20]. Heilman et. al [20] detect when a ROA has downgraded from valid to invalid or valid to unknown state and check whether a `.dead` object is present. This approach requires relying parties to perform ROVs, if it is to be effective. ROV deployment monitor [3], a monitoring platform [37] shows that only a few

[2]https://ftp.ripe.net/rpki/

[3]https://rov.rpki.net

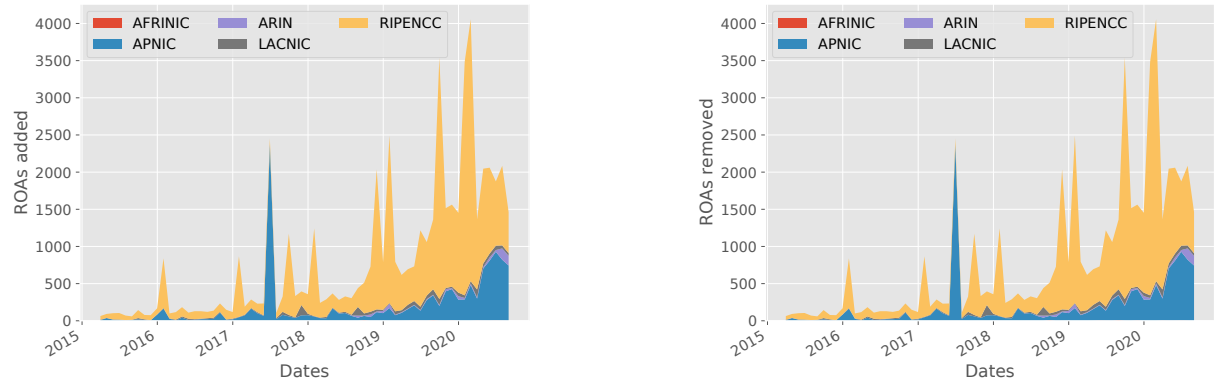

Fig. 8. Number of ROAs added and removed on average per day from March 2015 to August 2020

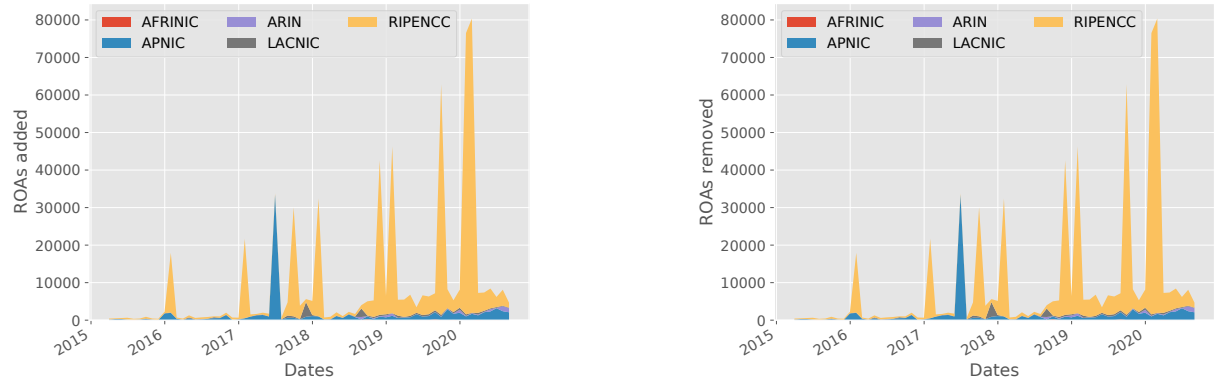

Fig. 9. Maximum per month of ROAs added and removed from March 2015 to August 2020

relying parties, 124 ASes, perform ROV, while there are 67599 ASes as of 10 August 2020 [4]. Furthermore, this approach fails in the hosted RPKI setting as the `.dead` objects that are used to signify consent from the child can be signed by the parent node allowing the parent to impersonate the child.

The second approach replaces the existing RPKI system with blockchain [19]. In this approach, the role of RIRs is limited to providing new resources and they cannot revoke already allocated resources. In addition to the large-scale changes that this approach requires, blockchain has other deployment issues such as consensus algorithm and the lack of incentive for the nodes to run the blockchain. While [34] suggested to use proof-of-stake as the consensus algorithm, it has the possibility to create another form of power imbalance where the nodes with a larger stake such as large ISPs will become powerful players.

VII. Conclusion

RPKI offers security benefits to BP and yet, it is not widely deployed. One reason is that it opens up the possibility for unilateral IP-prefix takedown. In this work, we make a small change to RPKI and propose a distributed RPKI that relies on prevention rather than detection of takedowns. As our solution requires communication between the RIRs, we hope that it will re-instigate discussions between the RIRs on the need for further collaboration. We propose two deployment models, the second of which eliminates the hierarchical structure of the existing RPKI and flattens the power relations. Both our deployment models distribute the trust anchor and prevent the scenario where validation fails due to the unavailability of a trust anchor. We perform extensive evaluation to assess the efficiency of our solution based on four threshold signature protocols and show that our solution scales when the deployment of RPKI increases.

[4]https://www.caida.org/data/as-relationships/

Acknowledgment

This work has been co-funded by: the German Federal Ministry of Education and Research and the Hessen State Ministry for Higher Education, Research and Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE; the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation): GRK 2050/251805230 and SFB 1119/236615297.

References

[1] ARIN. ARIN wins important legal case and precedent against fraud, 13 May 2019. https://www.arin.net/vault/about_us/media/releases/20190513.html.

[2] Hitesh Ballani, Paul Francis, and Xinyang Zhang. A study of prefix hijacking and interception in the internet. In *SIGCOMM*, pages 265–276. ACM, 2007.

[3] Donald Beaver. Efficient multiparty protocols using circuit randomization. In *CRYPTO*, volume 576 of *Lecture Notes in Computer Science*, pages 420–432. Springer, 1991.

[4] Henry Birge-Lee, Yixin Sun, Anne Edmundson, Jennifer Rexford, and Prateek Mittal. Bamboozling certificate authorities with {BGP}. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pages 833–849, 2018.

[5] Markus Brandt, Tianxiang Dai, Amit Klein, Haya Shulman, and Michael Waidner. Domain validation++ for mitm-resilient pki. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 2060–2076, 2018.

[6] Tim Bruijnzeels, Oleg Muravskiy, Bryan Weber, and Rob Austein. The RPKI repository delta protocol (RRDP). *RFC*, 8182:1–24, 2017.

[7] Kevin R. B. Butler, Toni R. Farley, Patrick D. McDaniel, and Jennifer Rexford. A survey of BGP security issues and solutions. *Proceedings of the IEEE*, 98(1):100–122, 2010.

[8] Christian Cachin and Asad Samar. Secure distributed DNS. In *International Conference on Dependable Systems and Networks, 2004*, pages 423–432. IEEE, 2004.

[9] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*, pages 136–145. IEEE Computer Society, 2001.

[10] Francisco Cifuentes, Alejandro Hevia, Francisco Montoto, Tomás Barros, Victor Ramiro, and Javier Bustos-Jiménez. Poor man's hardware security module (pmhsm): A threshold cryptographic backend for DNSSEC. In *LANC*, pages 59–64. ACM, 2016.

[11] Danny Cooper, Ethan Heilman, Kyle Brogle, Leonid Reyzin, and Sharon Goldberg. On the risk of misbehaving RPKI authorities. In *HotNets*, pages 16:1–16:7. ACM, 2013.

[12] David Cooper, Stefan Santesson, Stephen Farrell, Sharon Boeyen, Russell Housley, and W. Timothy Polk. Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile. *RFC*, 5280:1–151, 2008.

[13] Jim Cowie. China's 18-minute mystery, 18 November 2010. https://web.archive.org/web/20200109211935/https://dyn.com/blog/chinas-18-minute-mystery/.

[14] Anders P. K. Dalskov, Claudio Orlandi, Marcel Keller, Kris Shrishak, and Haya Shulman. Securing DNSSEC keys via threshold ECDSA from generic MPC. In *ESORICS (2)*, volume 12309 of *Lecture Notes in Computer Science*, pages 654–673. Springer, 2020.

[15] Ivan Damgård, Marcel Keller, Enrique Larraia, Valerio Pastro, Peter Scholl, and Nigel P. Smart. Practical covertly secure mpc for dishonest majority – or: Breaking the spdz limits. Cryptology ePrint Archive, Report 2012/642, 2012. https://eprint.iacr.org/2012/642.

[16] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 643–662. Springer, 2012.

[17] Data61. MP-SPDZ - Versatile framework for multi-party computation, 7 June 2019. https://github.com/data61/MP-SPDZ.

[18] Yossi Gilad, Avichai Cohen, Amir Herzberg, Michael Schapira, and Haya Shulman. Are We There Yet? On RPKI's Deployment and Security. In *NDSS*, 2017.

[19] Adiseshu Hari and T. V. Lakshman. The internet blockchain: A distributed, tamper-resistant transaction framework for the internet. In *HotNets*, pages 204–210. ACM, 2016.

[20] Ethan Heilman, Danny Cooper, Leonid Reyzin, and Sharon Goldberg. From the consent of the routed: improving the transparency of the RPKI. In *SIGCOMM*, pages 51–62. ACM, 2014.

[21] Amir Herzberg and Haya Shulman. Negotiating dnssec algorithms over legacy proxies. In *International Conference on Cryptology and Network Security*, pages 111–126. Springer, 2014.

[22] Amir Herzberg and Haya Shulman. Cipher-suite negotiation for dnssec: Hop-by-hop or end-to-end? *IEEE Internet Computing*, 19(1):80–84, 2015.

[23] Amir Herzberg, Haya Shulman, and Bruno Crispo. Less is more: cipher-suite negotiation for dnssec. In *Proceedings of the 30th Annual Computer Security Applications Conference*, pages 346–355, 2014.

[24] Tomas Hlavacek, Amir Herzberg, Haya Shulman, and Michael Waidner. Practical experience: Methodologies for measuring route origin validation. In *DSN*, pages 634–641. IEEE Computer Society, 2018.

[25] Geoff Huston, George Michaelson, and Robert Loomans. A profile for X.509 PKIX resource certificates. *RFC*, 6487:1–32, 2012.

[26] Geoff Huston, Mattia Rossi, and Grenville J. Armitage. Securing BGP - A literature survey. *IEEE Communications Surveys and Tutorials*, 13(2):199–222, 2011.

[27] Geoff Huston, Samuel Weiler, George Michaelson, Stephen T. Kent, and Tim Bruijnzeels. Resource public key infrastructure (RPKI) trust anchor locator. *RFC*, 8630:1–11, 2019.

[28] Jonathan Katz, Ueli Maurer, Björn Tackmann, and Vassilis Zikas. Universally composable synchronous computation. In *TCC*, volume 7785 of *Lecture Notes in Computer Science*, pages 477–498. Springer, 2013.

[29] Matt Lepinski, Andrew Chi, and Stephen T. Kent. Signed object template for the resource public key infrastructure (RPKI). *RFC*, 6488:1–13, 2012.

[30] Matt Lepinski and Stephen T. Kent. An infrastructure to support secure internet routing. *RFC*, 6480:1–24, 2012.

[31] Matt Lepinski and Kotikalapudi Sriram. BGPsec protocol specification. *RFC*, 8205:1–45, 2017.

[32] Charles Lynn, Stephen T. Kent, and Karen Seo. X.509 extensions for IP addresses and AS identifiers. *RFC*, 3779:1–27, 2004.

[33] M. Mueller, M. van Eeten, and B. Kuerbis. In important case, RIPE-NCC seeks legal clarity on how it responds to foreign court orders, 23 November 2011. https://www.internetgovernance.org/2011/11/23/in-important-case-ripe-ncc-seeks-legal-clarity-on-how-it-responds-to-foreign-court-orders/.

[34] Jordi Paillisse, Miquel Ferriol, Eric Garcia, Hamid Latif, Carlos Piris, Albert Lopez-Bresco, Brenden Kuerbis, Alberto Rodríguez-Natal, Vina Ermagan, Fabio Maino, and Albert Cabellos. IPchain: securing IP prefix allocation and delegation with blockchain. In *iThings/GreenCom/CPSCom/SmartData*, pages 1236–1243. IEEE, 2018.

[35] Qrator Labs. This is how you deal with route leaks, 2 April 2020. https://habr.com/en/company/qrator/blog/495260/.

[36] Raksha Reddy and Carl Wallace. Trust anchor management requirements. *RFC*, 6024:1–14, 2010.

[37] Andreas Reuter, Randy Bush, Ítalo Cunha, Ethan Katz-Bassett, Thomas C. Schmidt, and Matthias Wählisch. Towards a Rigorous Methodology for Measuring Adoption of RPKI Route Validation and Filtering. *ACM SIGCOMM Computer Communication Review*, 48(1):19–27, January 2018.

[38] RIPE NCC. The RIPE NCC's case against the state of the netherlands dismissed, 14 February 2013. https://www.ripe.net/publications/news/about-ripe-ncc-and-ripe/ripe-nccs-case-against-the-state-of-the-netherlands-dismissed.

[39] RIPE NCC. Accidental ROA Deletion, 2 April 2020. https://www.ripe.net/support/service-announcements/accidental-roa-deletion.

[40] RIPE NCC. A First for the RIPE NCC: Seizure of the "Right to Registration of IPv4 Addresses" for the Recovery of Money, 2 October 2020. https://labs.ripe.net/Members/ciaran_byrne/seizure-of-the-right-to-registration-of-ipv4-addresses.

[41] RIPE NCC. Closure of Members, Deregistration of Internet Resources and Legacy Internet Resources, 29 March 2019. https://www.ripe.net/publications/docs/ripe-716.

[42] RIPE NCC. RIPE NCC blocks registration in RIPE registry following order from dutch police, 9 November 2011. https://www.ripe.net/publications/news/about-ripe-ncc-and-ripe/ripe-ncc-blocks-registration-in-ripe-registry-following-order-from-dutch-police.

[43] Stephen M. Ryan and Sam C. Neel. Taking a hard line on fraud, 13 May 2019. https://teamarin.net/2019/05/13/taking-a-hard-line-on-fraud/.

[44] Kris Shrishak and Haya Shulman. Limiting the power of RPKI authorities. In *ANRW*, pages 12–18. ACM, 2020.

[45] Haya Shulman and Michael Waidner. Towards security of internet naming infrastructure. In *European Symposium on Research in Computer Security*, pages 3–22. Springer, 2015.

[46] Haya Shulman and Michael Waidner. One key to sign them all considered vulnerable: Evaluation of {DNSSEC} in the internet. In *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, pages 131–144, 2017.

[47] Mark Tinka. Rpki rov & dropping of invalids – africa, 09 April 2019. https://www.mail-archive.com/apops@apops.net/msg00796.html.

[48] Christopher S Yoo and David A Wishnick. Lowering legal barriers to rpki adoption. *U of Penn Law School, Public Law Research Paper*, (19-02), 2019.