Impacting IP Prefix Reachability via RPKI Manipulations

Kyle Brogle, Stanford University
Danny Cooper, Boston University
Sharon Goldberg, Boston University
Leonid Reyzin, Boston University

January 4, 2013

Abstract

The RPKI is an infrastructure that will provide digitally signed attestations for the hierarchical allocation and suballocation of IP addresses. Its goal is to improve security of interdomain routing by providing reliable data showing which autonomous system (AS) is authorized to originate which IP prefix. We discuss how the hierarchical nature of the RPKI makes it technically possible for any party above a target IP prefix in the RPKI hierarchy to revoke that target IP prefix. We show that such revocation can be "surgical"—i.e., impacting only the desired IP address or prefix—and difficult to detect. We also discuss the impact such revocation has on routing. This note focuses only on the issues of technical feasibility (rather than legal or operational issues), and should not be taken as recommendation for or against the use of the RPKI.

Contents

1 Introduction

The RPKI (see RFC 6480) is an infrastructure to support improved security for BGP routing. It enables an entity to verifiably assert that it is the legitimate holder of a set of IP prefixes, as well as to authorize an AS to originate an IP prefix in BGP. The information in the RPKI can be used to blunt the impact of the most common types of attacks on routing, namely prefix hijacks and subprefix hijacks [9, 22, 26].

IP address allocation generally operates in a hierarchical manner. IANA sits at the root of the hierarchy, allocating IP prefixes to the Regional Internet Registries (RIRs).[1] The RIRs then delegate subsets of their address space to national/local internet registries (NIRs or LIRs) or ISPs, who can then further delegate subsets of the space to other ISPs or customers. The RPKI adds cryptographic signatures to this hierarchy: it enables parties to digitally sign statements which indicate that they are delegating IP prefixes, as well as statements which indicate that they are authorizing ASes to originate IP prefixes (in both cases, the signing party, unless it is a root of the hierarchy[2], would also have to show that it was delegated this IP prefix through the RPKI).

Recently, however, concerns have been expressed that, while RPKI eliminates some risks, it may introduce new ones [4, 25]. The RPKI allows delegators of address space to alter or revoke their delegations (by modifying certificates and employing certificate revocation lists). This power is unilateral, not requiring any action on the part of the recipient. Therefore, RPKI may provide the delegators of address space with a technical means to easily and unilaterally revoke it in a manner that could affect prefix reachablity in BGP. The concern is that this power can be abused.

The goal of this note is to provide a technical analysis of this concern. We show that it is technically feasible for any party to invalidate a target IP-prefix-to-AS mapping contained in *any* certificate below it in the RPKI hierarchy. Moreover, this action can be done in a 'surgical' manner, in the sense that it need not invalidate any IP-prefix-to-AS mapping other than that of the target. We also discuss the nuances involved in determining when the invalidation of an IP-prefix-to-AS mapping in the RPKI can harm prefix reachability in BGP.

1.1 Motivation and context

At a very high level, we are interested in the following question: can one party (*the revoker*) use RPKI to unilaterally impact the reachability of a target IP prefix? This note focuses only on the technical issues behind this question: we consider what is possible within the RPKI specification. We do not assess whether some revocations would violate ICANN rules, registries' agreement with their constituent members, business contracts, or local laws (while such an assessment would be valuable, we note that legal limits are frequently ignored in the face of political and other pressures and, moreover, can be modified locally by new laws).

Concerns expressed in the past [4, 25] were focused mainly on RIRs and NIRs being obliged by their governments to revoke certificates in response to "takedown" requests, raising significant questions of policy, economics, and civil rights. Indeed, there is a risk that a government may use lawful process or extralegal pressures to force an RIR, NIR, or LIR within its jurisdiction to use the RPKI to revoke IP addresses that host content found to be undesirable or illegal, notwithstanding the registries contractual obligations with ICANN and their constituent members.

[1] In a few legacy cases, IANA bypassed RIRs and allocated prefixes directly.

[2] The roots of the RPKI hierarchy are not specified, but will likely be the RIRs or IANA—see RFC 6480 Section 2.4

We broaden this question to consider unilateral revocations by *any* delegator of IP prefixes in the RPKI, not just the RIRs. Indeed, extending Amante's apt comparison of an RIR to a registry of deeds [4] for real estate, we can think of the entire hierarchy as a system of leases and subleases. The question, then, is whether a landlord has unilateral power to evict a tenant or a subtenant, thus creating a precisely the imbalance of power that eviction laws in many jurisdictions try to correct by requiring a judicial process.

In broadening this question, we emphasize that our concern is not only in takedowns by RIRs and LIRs at the behest of governments. As transfers and leases of IPv4 prefixes become more common (facilitated, in part, by the RPKI), business disputes are likely to arise. For example, an ISP A may wish to take back address space that it leased to ISP B because the lease has expired, or because B has reneged on a payment. Alternatively, A may want to violate its contract with B and terminate the lease early for financial reasons or because a subtenant of B is hosting a website that is critical of A. Note that A does not even have to be a provider of B, only the delegator of IP prefixes. The RPKI will provide A with the technical means to cheaply and quickly revoke the address space of B regardless of who is in the wrong, whereas in today's world without RPKI, A would be forced to negotiate or litigate to settle the dispute. Note that the RPKI does not provide a similar power to B: the recipient of the resource cannot do anything similar to counter the delegator's move.

Of course, revoking address space within the context of RPKI is not the only way to make a resource unreachable on the Internet. But the existence of other vulnerabilities should not prevent us from studying this one. Moreover, typical censorship methods used today are less devastating. For instance, the commonly-used forms of intra-AS or intra-country censorship (*e.g.,* TCP resets, infinite HTTP redirect loops, traffic interception, or interception and overwriting of DNS responses [1, 11]) are intended to impact only the Internet users inside the censoring AS or country. DNS takedowns — when records are removed from DNS nameservers — have a more global impact, preventing a target domain name from being resolved to its IP address by users *anywhere* in the Internet.[3] However, during a DNS takedown, the target is still reachable by its IP address and, moreover, may find a DNS server in another jurisdiction to host its record.

In contrast, RPKI invalidation is a routing-based method and thus has the potential to completely prevent an IP prefix from being reachable by large swaths of the Internet (we discuss this in more detail in Section 3). In fact, we have seen routing-based blocking methods used in practice on rare occasions [25]. And blocking through RPKI manipulation has the potential to be even more powerful than other routing-based methods available today:

- Filtering at the recipient AS end or at some intermediate AS will leave the target accessible in other parts of the Internet and, therefore, reachable via Internet proxies.

- A provider may be convinced to stop routing for its customer, but a multihomed customer or a customer with provider-independent address space will have other options.

- Another routing-based blocking method is to force disconnection through the use of Internet Routing Registries (IRRs), `http://www.irr.net/`, which contain information on which AS is authorized to originate a given IP prefix and are used by many ISPs to filter routing announcements. However, accomplishing disconnection through IRRs is difficult, because

[3]There are known cases DNS takedowns *e.g.,* [10], and these cases may become more common if proposals like SOPA [27] become law.

(a) not all ISPs use IRRs to construct filters, and (b) IRR data is known to be stale and inaccurate, so even those ISPs who use IRR data typically take the union of some subset of the IRR repositories (there are currently 36 of them, `http://www.irr.net/docs/list.html`). This means that to affect reachability, the target prefix must be removed from most IRR repositories, which is made even more difficult by the fact that many IRR repositories are located in different jurisdictions. In contrast, RPKI repositories will all change in response to the actions of a single party—the revoker.

1.2 Overview of results

Invalidating a target prefix in RPKI. In Section 2 of this note, we show in detail how a *revoker* can make any *target* IP prefix that it has delegated *invalid* in the RPKI. By itself, this is not surprising, because, after all, RPKI has provisions for revocation of delegations. What makes our results interesting is the following:

1. A revoker can invalidate delegations through small and difficult to detect changes to the RPKI, and without explicitly employing certificate revocation lists. To achieve this, a revoker can

   (a) Delete a certificate it issued from its RPKI publication point.

   (b) Overwrite a certificate it issued with a different one for smaller set of resources.

   (c) Let a certificate it issued expire and refuse to renew it.

   (d) Add a new certificate to cause certain address space to become invalid. (This observation was also made by [28] and others.)

   We discuss these actions in detail in Sections 2.2.1 and 2.2.4.

2. The revoker can accomplish revocation regardless of how far below in the allocation hierarchy the target prefix is actually utilized. We give some examples of this in Sections 2.2.2 and 2.2.3, and discuss a generic procedure for doing this in Section 2.3.

3. Revocation can be done with any granularity, regardless of how big a chunk of address space was initially delegated. See Section 2.3.

4. Revocation can be "surgical" and does *not* have to do collateral damage to the *validity* of other address space, even if the target prefix is part of a bigger delegated chunk, or if the target prefix is far below the revoker in the allocation hierarchy. See Section 2.3. While the revoker's actions will not cause any other address space to become invalid, these actions may, in some cases, require significant refactoring of the RPKI, and therefore may be detectable by monitors; we discuss this issue throughout Section 2 and especially in Sections 2.4 – 2.5.

Impacting prefix reachability via invalidation in the RPKI. It is important to note that invalidating a target prefix in the RPKI does *not* necessarily mean that it becomes unreachable in BGP. The reachability of a prefix depends strongly on how RPKI information is consumed by relying parties (*i.e.,* parties other than the revoker and the owner of the target prefix). We study these issues in Section 3. Because the RFCs do not specify exactly how parties should consume RPKI information (instead, they state that this is a matter of "local policy" at the relying parties—see RFC 6483 Section 3), we instead consider a number of plausible local policies for consuming

RPKI information, and discuss their impact on prefix reachability. We consider using RPKI to (a) allow a provider to decide whether or not to originate a prefix for its customer (Section 3.1), (b) allow a provider to construct prefix filters for its stub customers (Section 3.2), and (c) allow a router to perform origin authentication (Section 3.3). We find the while RPKI invalidation can affect prefix reachability to different degrees in each of these three cases, the impact in all cases is typically quite nuanced. The only exception is the case where routers use a strict "drop invalid" origin authentication policy, *i.e.,* they refuse to accept invalid routes, so that invalidating a target prefix unequivocally prevents it from being reachable via BGP (Section 3.3.2).

Recommendations. We suggest that further study is required into techniques for detecting RPKI manipulations and minimizing their impact on routing while still preserving the security benefits promised by the RPKI. We provide some preliminary discussion of this in Section 2.5.

2 Surgically invalidating a target prefix

In this section, we show how a holder of IP addresses in RPKI can invalidate any (IP prefix, origin AS) pair where the prefix is a subset of the address space it owns. This action is "surgical", in the sense that all other (IP prefix, origin AS) pairs that were valid according to the RPKI prior to the revocation also remain valid afterward. Moreover, this action can be taken at any depth of the RPKI certificate hierarchy.

Organization. We start with background on RPKI in Section 2.1, give examples of surgical revocations in Section 2.2, present a generic procedure for surgical revocation in Section 2.3, and finally discuss the detectability of these procedures in Section 2.4 and Section 2.5.

2.1 RPKI Background.

We highlight with a few essential points from RPKI and introduce some terminology and schematics that will simplify the presentation:

1. **Allocations and Resource certificates (RCs).** In RPKI, every holder of resources (for our purposes, IP prefixes) who can suballocate them to others or authorize their use is a certificate authority (CA) (see RFC 6480 Section 2.2). Every CA has a resource certificate (RC) showing which IP prefixes it holds. If CA A wants to sub-allocate some address space to B, it issues an RC for B for that address space; for this RC for B to be valid, the set of addresses it contains must not contain any addresses that are not covered by the RC of A, and, of course, the RC of A itself must be valid (see RFC 6487 Section 7. When A issues an RC to B, A is the *issuer* and B is the *subject* of the RC. To simplify the discussion, we also say that A is the *parent* of B, while B is the *child* of A. We can extend this definition to grandparent, grandchild, great-grandparent, great-grandchild, *etc.* in the obvious way.

2. **Route Origin Authorizations (ROAs).** RCs by themselves do not provide authorizations to route. To authorize an AS to originate one or more prefixes, a CA issues a one-time-use end-entity (EE) certificate, which is then used to sign a message called "Route Origination Authorization" (ROA) that contains the origin AS number and the prefixes. Since each EE certificate is used to sign a single ROA and is stored as part of that ROA (RFC 6480 Section 2.3), to simplify the discussion, we will generally ignore EE certificates. For example,

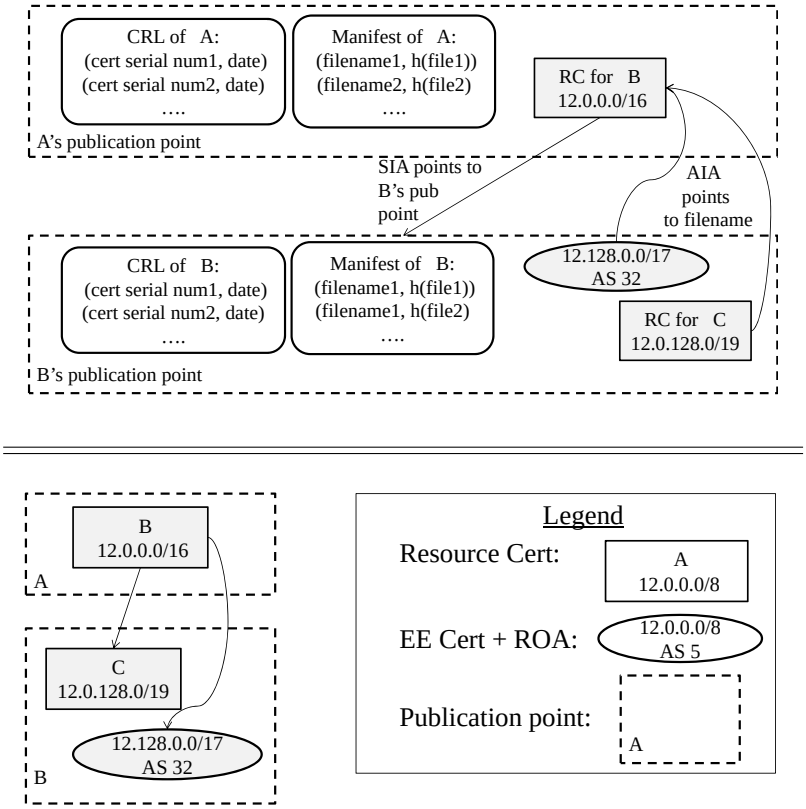

Figure 1: The top part shows the schematic of RPKI directory structure. The bottom part shows the simplified version on the left and the legend on the right.

suppose ROA r is signed by EE certificate that was issued by a CA B. We will simply say that ROA r was issued by B. For a ROA to to be valid, the set of addresses covered by the ROA must be covered by the RC of its issuing CA (RFC 6482 Section 4). A ROA for AS 0 means "do not route," although it can be superseded by another ROA with a valid AS (RFC 6483 Section 4). Note that it is possible to have a ROA for an IP prefix (for example, 12.0.0.0/17) and another ROA for a subset of that prefix (for example 12.0.0.0/18); in BGP, the more specific prefix (i.e., the subset) will take priority. A ROA has a maxLength field to prevent ASes from advertising prefixes that a more specific than they are authorized (*e.g.,* 12.0.0.0/17 can have a maxLength of 17, to prevent the corresponding AS from advertising 12.0.0.0/18). RCs have no maxLength field and thus no mechanism to prevent delegation of arbitrarily specific prefixes.

3. **Repository structure.** RCs and ROAs are placed in a repository (RFC 6481 Section 2), which is a structure of directories and files, each of which can be identified by a universal resource identifier (URI). Each RC and ROA is in its own file with its own filename. An updated RC or ROA may have the same filename as the previous one—the old version in the repository simply gets overwritten (see RFC 6481 Section 2.2). Each RC and ROA (except the root RC, of course) contains the URI for filename of the RC for the CA who issued it, so

that certification chains can be traced up the hierarchy (this URI is stored in the so-called "Authority Information Access (AIA)" field; we prefer to call it "parent pointer"). Everything signed by CA B—that is, RCs, ROAs, and the certificate revocation lists (CRLs) issued by B—are placed in a single directory (equivalently, folder, or as the RFCs call it, *publication point*) for CA B. The RC of B, which is stored in the directory of his parent, contains the URI of the directory where B will place everything it signs (it is stored in the so-called "Subject Information Access (SIA)" field; we prefer to call it "child pointer"). Thus, all the pointers reflect the tree structure of the certificate hierarchy, with every node pointing to the file of its parent and to the directory of its children. These details are illustrated in Figure 1, which also shows the simplified trees we will be using later.

4. **Manifests.** To provide assurance that no items have been deleted from the repository, the hash values for all the files issued by B are collected into a single file called *manifest* (RFC 6481 Section 2.1), which is digitally signed by B and stored in B's publication point. (See Figure 1.) This file needs to be updated whenever B issues, modifies, or revokes an RC or a ROA. In order to avoid complicating the presentation, we will not address manifests explicitly—we will simply assume that each CA updates its *own* manifest as needed, which it can easily do.

5. **Certificate revocation lists (CRLs).** The parent of an RC or the issuer of a ROA can revoke it simply by adding it to his CRL. (In the case of a ROA, it is actually the corresponding EE certificate that will be added to the CRL, but that doesn't make a difference for our purposes, since EE certificates and ROAs are in one-to-one correspondence.)

We should note that the RPKI also allows for certifying allocation of AS numbers in the same way as IP addresses, and will likely be used to issue end-entity certificates for AS numbers for use in BGPSEC [18], although this feature will generally not concern us. It is important for us, however, that each RC must contain at least one IP prefix or AS number; an RC that has no resources is not allowed.

2.2 Invalidation examples

We start by presenting some example of surgical invalidation when the target ROA is issued by the revokers itself (Section 2.2.1), by its child (Section 2.2.2), and by its grandchild (Section 2.2.3). These examples refer to the certificate hierarchy in Figure 2. The target will always be AS 13, and the revokers will be either F, B, or A. In Section 2.3 we show a general invalidation procedure for any descendant. We also discuss how adding a new ROA can cause address space to become invalid (Section 2.2.4).

2.2.1 Invalidating address space in a ROA issued by the revoker

Scenarios. The RPKI envisions a scenario in which customers would get ROAs issued by their providers. Specifically, RFC 6480 Section 7.3.2 considers a multi-homed customer who is assigned address space by its primary ISP; the customer has an AS number and is multihomed, advertising routes through its multiple providers. A recommended was to handle this scenario is for the primary ISP to issue a ROA authorizing the customer's AS number to originate the assigned address space.

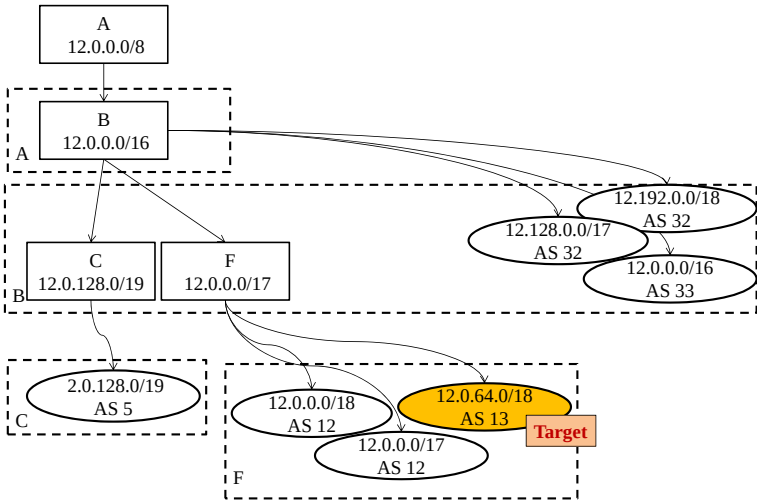

Figure 2: Sample RPKI.

Referring to Figure 2, suppose the multi-homed customer is AS 13, its primary ISP is F, and its assigned address space is 12.0.64.0/18.

Note that F cannot prevent 12.0.64.0/18 by being announced by AS 13 simply by prefix filtering, because AS 13 is multi-homed. However, F may be able to prevent 12.0.64.0/18 (or some subset of that address space) from being reachable at AS 13 through RPKI invalidation.

Invalidating the entire prefix. There are a number of ways F can invalidate the *entire* 12.0.64.0/18 prefix given to AS 13. In the following, we refer to 12.0.64.0/18 as the *target prefix*, and the ROA mapping 12.0.64.0/18 to AS 13 as the *target ROA*.

1. F can simply add the target ROA (to be precise, the EE certificate in the ROA) to its CRL. This action is quite obvious, because the CRL is signed by F and published in the repository.

   Note that F can also take more subtle actions:

2. F can simply let the ROA quietly expire, if F is willing to wait until the expiration time.

3. F can delete the ROA from his publication point (and also from the manifest; since the manifest is managed and signed by the revoker, it can simply issue a new signed manifest that does not include the deleted ROA).

4. F can *overwrite* the target ROA; that is, replace the target ROA with another one for a different AS number (e.g., F's AS number, or AS 0 —see RFC 6483 Section 4) but the same filename. We note that a non-normative guideline in RFC 6481 Section 2.2) suggests deriving the ROA filename from the name of the public key of the EE certificate; thus, if F had the foresight to store the private key that corresponds to the public key in the target ROA's EE certificate, F can issue the new ROA (mapping the target prefix to, say, AS 0) using this private key. In this way, F achieves the non-normative requirement that the filename for the new ROA will is the same as the filename for the original target ROA, and overwrites the target.

Deleting or overwriting the ROA may have the same effect on the relying parties as adding it to the CRL, but may be more difficult to detect. Indeed, the recommended method for accessing the repository, rsync (see RFC 6481 Section 3), and the leading relying party software rcynic [2] make it easy to synchronize the local copy of the RPKI at the relying party to entire directory trees, obtaining not only the latest CRL, but also deleting certificates that are no longer in existence, updating ones that have been modified, and obtaining newly issued ones.

Invalidating a portion of a prefix. We note that F can also invalidate a *portion* of the 12.0.64.0/18 prefix, by issuing a single new ROA containing the prefixes that cover the parts of 12.0.64.0/18 that F wants to remain valid (a single ROA can authorize many prefixes RFC 6482 Section 3). The new ROA can replace the old one in the repository if it is given the same filename (using the same overwriting process described in item 4 above); alternatively, the old ROA can be invalidated using any of the methods described in items 1-3 above.

Detectability & a recommendation. Note that the only impact of this action on the RPKI is the expiration, deletion, or change of one ROA. Given the natural churn of ROAs, including routine reissuance due to expiration, it is unlikely that such a change will be noticed unless someone is looking for it. Indeed, ROAs will be naturally routinely deleted from the repository when they are close to expiration, because reissued ones will replace them (depending on the naming scheme used, the new ones may have the same or different filename). However, it might be helpful for future versions of relying party software like rcynic to detect when ROAs have been deleted and overwritten, especially when they have been overwritten with a ROA that certifies a smaller portion of the address space, or a ROA for AS 0.

Does having my own RC reduce the risk of ROA invalidation? One might wonder if AS 13 might avoid having its ROA invalidated if it had its own RC. For instance, in our example, AS 13 might be the holder of an RC for its prefix 12.0.64.0/18. Indeed, returning to the scenario described above, note that RFC 6480 Section 7.3.2 suggests that, alternatively, a multihomed customer can get an RC from its primary ISP, and issue its own ROA.

However, in the next section, we show that obtaining your own RC does not provide much protection against invalidation, either.

2.2.2 Invalidating address space in a ROA issued by the revoker's child

Refer to Figure 2, and now consider the case when the revoker is B, and the target ROA is still the same, issued by B's child F authorizing AS 13 to originate prefix 12.0.64.0/18.

Scenarios. Perhaps AS 13 and F are the same entity: a multi-homed subscriber. Its primary ISP is B, who allocated F's address space. Instead of getting a ROA direclty from its primary ISP B, F chooses the other alternative recommended by RFC 6480 Section 7.3.2: B issues an RC for F, who issues its own ROA for its own AS number. We will show that this does not help much in protecting F from RPKI invalidation.

Alternatively, the same RPKI scenario may arise when F has provider-independent address space, allocated by B (RFC 6480 Section 7.3.3). In that case, B may or may not have anything to do with routing traffic for F; the business relationship between B and F may have ended a long time ago, and F is issuing its own ROAs. Or, in another alternative, AS 13 and F may be different entities: AS 13 may be a customer of F, and F got its allocation from B.

Invalidation procedure. We show how B can invalidate the entire 12.0.64.0/18 prefix (invalidating a portion of that prefix can be done similarly). Note that the target ROA, as well as ROAs

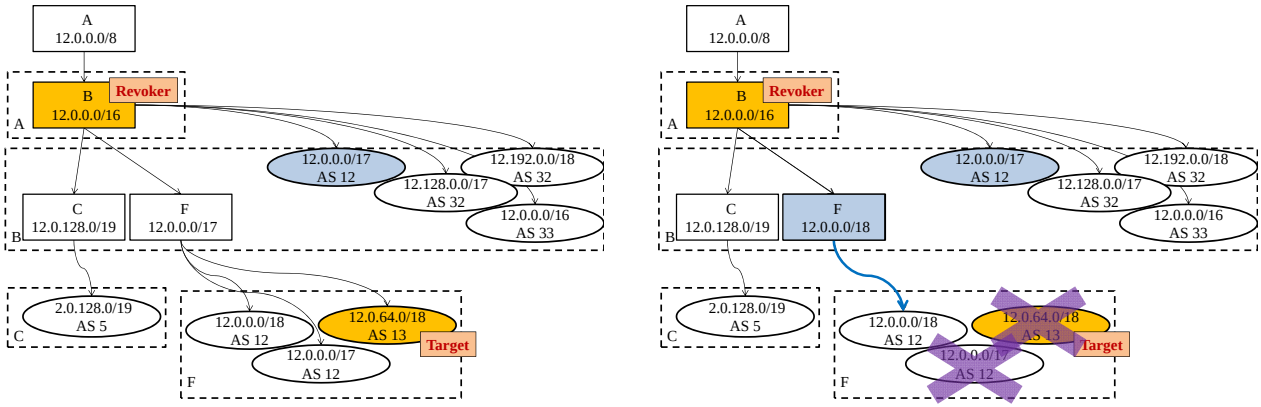

Figure 3: Invalidating a child's ROA.

that authorize AS 12 to originate 12.0.0.0/18 and 12.0.0.0/17, are issued by F and are therefore in F's directory in the repository, so B cannot simply delete or modify them. The revoker B proceeds as described below and shown in Figure 3:

1. **Prepare a new child RC.** The revoker prepares, but does not yet publish, a new RC for F that excludes the target IP prefix 12.0.64.0/18. The new RC for F is for the prefix 12.0.0.0/18, and includes the same public keys, parent and child pointers, and subject name as the original RC for F.

2. **Grandparent the impacted ROAs.** The revoker reissues the ROAs for F that are not covered by the new RC for F, except for the target (IP prefix, AS) pair, if any such ROAs exist. In this example, the ROA mapping 12.0.0.0/17 to AS 12 is not covered by the new RC for F, and is therefore reissued by the revoker and placed in the revoker's own directory. See the left side of Figure 3.

3. **Overwrite the child RC.** Finally, the revoker replaces the original RC that gives 12.0.0.0/17 to F with the RC prepared in Step 1, by publishing the new RC in the repository with the same filename as the old one. This causes the target ROA to become invalid. This also causes the ROA issued by F that authorizes AS 12 to originate 12.0.0.0/17 to become invalid; fortunately, a ROA for the same (prefix, origin AS) pair has been reissued by the revoker and placed in the revoker's directory, and so the RPKI continues to indicate that AS 12 is a valid origin for prefix 12.0.0.0/17. Finally, F's original ROA for 12.0.0.0/18 remains valid; this follows because the new RC for F contains the same public key and subject identifier as the original RC for F, and moreover has the same filename as F's original RC. See the right side of Figure 3.

This example illustrates that a child's ROA may be "cleanly" invalidated, with the only impact to the RPKI being that F can no longer issue ROAs that are not covered by his newly issued RC (in this example, the ROA binding 12.0.0.0/17 to AS 12). However, the (prefix, origin AS) pairs in these impacted ROAs remain valid according to RPKI for the duration of the revoker's actions, and therefore this action has no impact on the reachability of these prefixes.

Detectability & Grandparenting. Note that in the example above, the revoker never actually

revokes any certificates; instead, it *overwrites* an old RC with a new one for a smaller set of resources, and issues a few new ROAs.

However, note that step 2 leaves the RPKI is a somewhat strange state. Namely, there are two copies of the ROA for prefix 12.0.0.0/17 — one stored in F's publication point, and another stored at B's publication point. We note, however, the new ROA published by B will still be valid: RFC 6488 Section 3 states that a ROA is valid as long as "a valid certification path from a trust anchor to this [ROA] exists," which is certainly the case for the new ROA. We note further, that the RPKI was specifically designed *not* to require uniqueness of certificates for resources, in order to allow resource holders to adhere to the principle of "make before break" (see RFC 6480 Section 7.3), which states that a new certificate should be issued *before* the old one is set to expire or be revoked. (The make-before-break principle means, for example, that when the originating AS for a prefix changes, a ROA for the new AS is issued before the ROA for the old AS expires or is revoked; or that, during a key rollover RFC 6489, RC for the new key, as well as RCs signed by that key, are issued before the old key is set to expire; or that, when an address block is transferred between RIRs, the RC assigning the address space to the new RIR is issued before the RC assigning the space to the old RIR is set to expire.)

Note that that the draft Responsible Grandparenting in the RPKI [6] refers to the situation in which a certificate (in our case, ROA for prefix 12.0.0.0/17) is reissued by its grandparent's RC (in our case, B) instead of its parent's RC (in our case, F) as "grandparenting." At the time of this writing, there is some debate as to whether "grandparenting" should be a recommended or even an allowed practice in the RPKI. Some argue that grandparenting is operationally useful: for example, [6, Section 1] argues that "there are circumstances in RPKI operations where a resource holder's parent may not be able to, or may not choose to, facilitate full and proper registration of the holder's data", and provides various examples for when this might occur, for instance when a parent that is going out of business. On the other hand, others argue that if the grandparent does not have a contractual relationship with the grandchild, it should not be issuing ROAs for it.

Regardless of whether grandparenting is ultimately accepted as a legitimate or recommended practice, it is important to emphasize that the grandparenting action is technically possible within the RPKI specification, because the ROA issued by the grandparent will be considered "valid" according to RFC 6488 Section 3. If it is ultimately decided that grandparenting is a legitimate practice, then the revoker's actions might be difficult to distinguish from normal churn in the RPKI. On the other hand, if grandparenting is ultimately discouraged, then this step of the revocation procedure could make the revoker's actions more easy to detect. However, if there is a concern that grandparenting may make B's action detectable, B may also be able to convince F to issue those ROAs instead—see Section 2.4.

Grandparenting is sometimes not needed... We should note that if the example is modified so that the target prefix is exactly covered by a ROA, and no more general prefix is contained in any other ROA issued by F (i.e., the 12.0.0.0/17 ROA for AS 12 does not exist), then the revoker B does not even need to issue any ROAs. That is, no ROA grandparenting is required, which makes this action even more difficult to detect. There is one caveat in this case: if the RC issued to F has the same set of resources as the ROA for AS 13 (i.e., the are no other ROAs below the RC of F), then the revoker also does not need to issue any new ROAs. However, in such a case, the revoker has a slight difficulty: it may not be able to simply overwrite the RC of F with an RC for a smaller set of resources, because that smaller set may be the empty set, and RCs for an empty set of resources are not allowed in the RPKI per RFC 6487 Section 4.8.10 (and would be quite

conspicuous if published). There are at least five possible solutions to this problem: B can add F's RC to B's CRL; B can issue a new RC to F that has no IP resources but only some AS resources (assuming B has any AS resources); B can issue a very narrow RC for F—perhaps for a single IP address—that would severely limit F's options; B can delete F's RC in the repository; or B can let F's RC expire. The last two options, in particular, would probably not be easily distinguished from routine churn.

2.2.3 Invalidating address space in a ROA issued by the revoker's grandchild

Next, we consider the case where the target ROA was issued by the revoker's grandchild. Note that the revoker's relationship to the target may be very tenuous in this case. Invalidation becomes more complicated for the revoker.

In our example, the revoker is A, and the target ROA, as before, is issued by A's grandchild F, and authorizes AS 13 to originate prefix 12.0.64.0/18. Like in the previous example, the target ROA, as well as ROAs that authorize AS 12 to originate 12.0.0.0/18 and 12.0.0.0/17, are issued by F and are therefore in F's directory in the repository.

In the text below and in Figure 4, we show how A can invalidate the entire 12.0.64.0/18 prefix (invalidating a portion of that prefix can be done similarly).

1. **Prepare a new child RC.** The revoker prepares, but does not yet publish, a new RC for its child B. While the original RC for B was for 12.0.0.0/16, this new RC excludes the target prefix 12.0.64.0/18, and therefore includes only the prefixes 12.0.0.0/18 and 12.0.128.0/17. The subject name, parent and child pointers, and public key fields of the reissued RC remain the same.

2. **Grandparent the impacted child ROAs.** The new RC for B no longer covers B's ROA mapping 12.0.0.0/16 to AS 33, which will make this ROA invalid once B's new RC replaces the old one. Therefore, the revoker A now needs to reissue the ROA for (12.0.0.0/16, AS 33) in its own directory. See the top left of Figure 4.

3. **Grandparent the RC.** Next, the revoker issues a new RC for its grandchild F that excludes the target IP prefix 12.0.64.0/18 (since the old RC for F will no longer be valid once B's old RC is replaced with the new one, because it contains an address range not covered by its parent). The new RC for F is for the prefix 12.0.0.0/18, and includes the same public key, child pointer to F's directory, and subject name as the original RC for F. However, the parent pointer, instead of pointing B's RC, points to the revoker's RC. This new RC is published in the revoker's directory. See the top right of Figure 4.

4. **Grandparent the impacted grandchild ROAs.** All ROAs issued by F's old RC all have the parent pointer pointing to F's old RC. Since F's old RC will soon become invalid, these ROAs need to be reissued. For this reason, the revoker now reissues all of ROAs (except for the target ROA) that are descendants of F's old RC, making sure their parent pointers point to the revoker's RC, and publishes them in its directory. See the top right of Figure 4.

5. **Overwrite the child RC.** Finally, the revoker replaces the original RC that assigns 12.0.0.0/16 to B with the RC prepared in Step 1, by publishing the new RC in the repository with the same filename as the old one. This causes the original RC for F to become invalid, along with

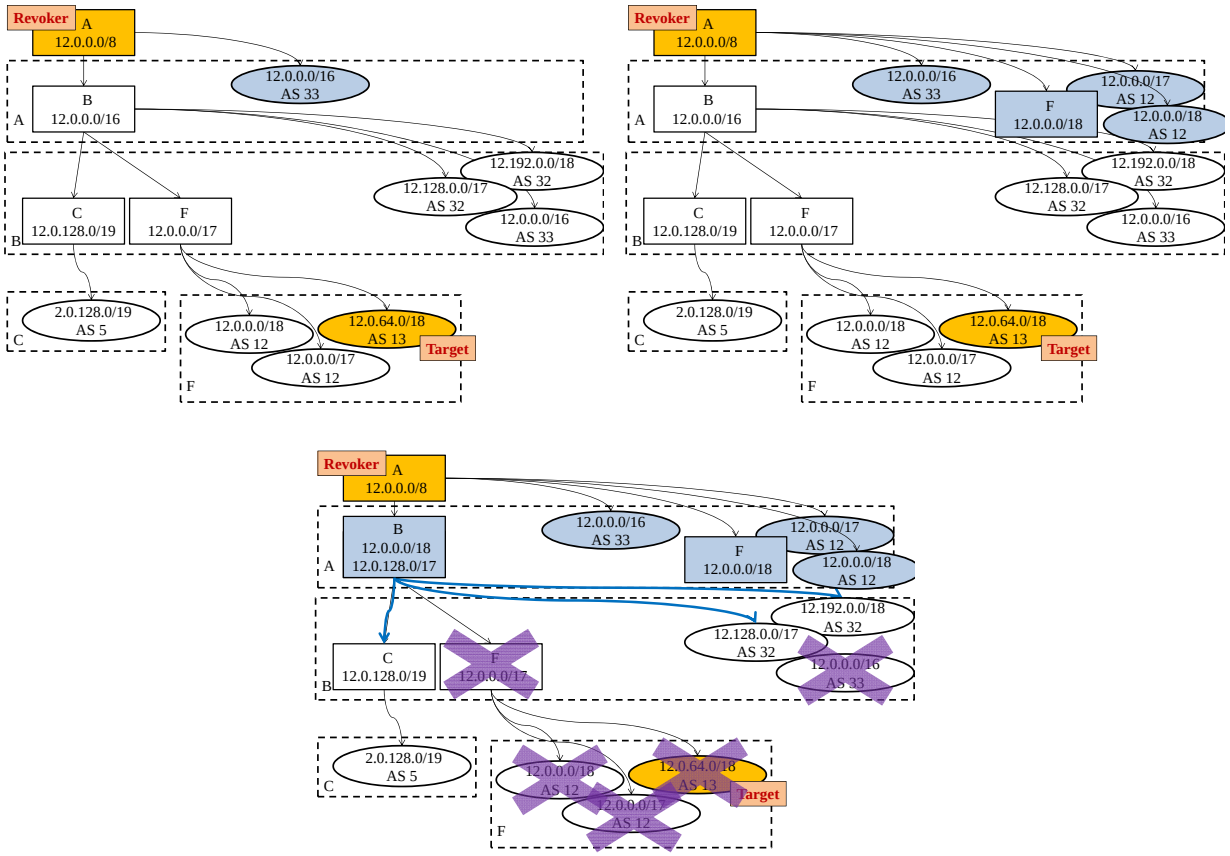

Figure 4: Invalidating a grandchild's ROA.

all the ROAs directly issued by F's original RC. Fortunately, however, all of these RCs and ROAs (except the target ROA) have been reissued by revoker A and placed in A's directory. We also note that C's RC (issued by B) remains valid, as it is covered by the new RC for B, which has the same filename as the original RC for B. Moreover, the original ROAs issued by B for prefixes 12.128.0.0/17 and 12.192.0.0/18 remain valid, because they are covered by the new RC for B. See the bottom of Figure 4.

This example illustrates that a grandchild's ROA may be invalidated, with the only impact to the RPKI being that the child and grandchild B can no longer issue ROAs that are not covered by their newly issued RC (in this example, the ROA binding 12.0.0.0/16 to AS 33 and the ROA binding 12.0.0.0/17 to AS 12). Note that, even though *all* the ROA's for the grandchild F have all been reissued by A after the invalidation, F still has a the ability to issue the ROAs covered by his new RC, because F's new RC has a child pointer to F's publication point (in this example, F can still issue the ROA binding 12.0.0.0/18 to AS 12). Note, however, the F now loses the ability to issue ROAs that are *not* covered by his new RC. Either way, all (prefix, origin AS) pairs in these reissued ROAs are still valid according to the RPKI for the duration of the revoker's action, and therefore these actions have no impact on the reachability any prefix other than the target prefix.

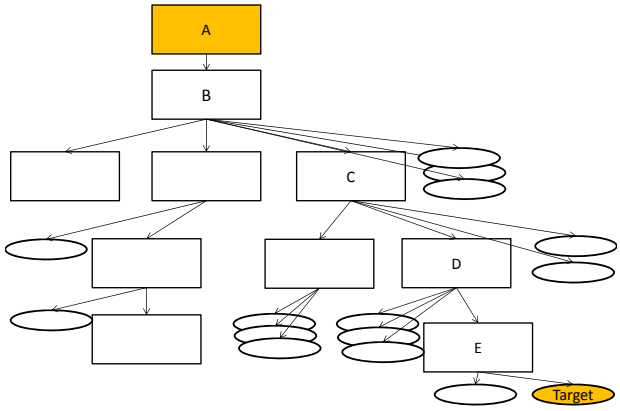

Figure 5: Generic procedure for surgical invalidation.

Detectability & Grandparenting. It should be noted that this invalidation creates an unusual repository state, because we have a variety of grandparented ROAs and a grandparented RC (the new repository has two RCs for the same public key of F: namely, the old one issued by B (no longer valid, because it is not covered by B's new address range), and the new one issued by A.) While this unusual state does not impact the validity of any ROAs except the target ROA, as we discussed in Section 2.2.2, it makes it easier to detect that something unusual is going on. Thus, even though no certificates were explicitly revoked, grandchild invalidation is likely more detectable than the previous examples. We do note, however, that the revoker can skip the step where he grandparents the RC (item 3 above) to make his actions more difficult to detect; the only impact of doing this is to remove F's ability to issue ROAs himself. This does not impact the validity of any (prefix, AS) pairs.

We discuss more ways for A to reduce detectability of its actions in Section 2.4.

2.2.4 Invalidating address space by adding a new ROA

The ROAs in the RPKI do not cover all the IP prefixes. Any prefix that is not present in a valid ROA and is not a subset of a prefix present in a valid ROA is considered "unknown" RFC 6483 Section 2. It is likely that routing for "unknown" prefixes will continue the same way as without RPKI.

However, as observed by [28], an "unknown" prefix can be made "invalid" by any revoker who has an RC covering the prefix. The revoker needs to simply issue a ROA for that prefix (or its superset) and any AS—an AS that does not exist, AS 0 RFC 6483 Section 4, itself, or any AS that is unlikely to originate that subprefix in BGP.

2.3 Generic procedure for surgical invalidation of a descendant's target ROA

We are now ready to present the generic procedure for surgical invalidation of an IP prefix (or its portion) contained in a ROA of a descendant. (Note that there is no reason to require that the prefix is actually contained in an issued ROA—it could be contained only in an RC, and the procedure would remain the same.)

Suppose a revoker holding an RC A wants to invalidate an IP prefix that is contained in a target ROA t that is below RC A in the RPKI hierarchy. Let B denote the child of A and C denote the grandchild of A on the path to target t (see Figure 5). The revoker proceeds as follows:

1. **Prepare a new child RC.** Prepare, but do not yet publish, a new RC for A's child B, that excludes the target IP prefix(es).

2. **Grandparent the impacted child ROAs.** Consider all the ROAs directly issued by the original RC of B. Of these ROAs, reissue only the ROAs whose addresses are not covered by the new RC of B, because they will become invalid once the new RC replaces the old one.

3. **Grandparent the RC.** Since the RC of the grandchild C will have addresses not covered by the new RC of B, it will become invalid one the new RC for B replaces the old one. Therefore, issue and publish (in A's directory) a new RC for C that excludes the target IP prefix(es).

4. **Grandparent all the RCs and ROAs below grandchild's RC.** Since ROAs and RCs of C and its descendants will all become invalid once the new RC for B is published (because, by virtue of parent pointers, their verification will go through the old RC for C, which will become invalid because its addresses will no longer be covered by its parent's addresses), reissue *all* RCs and ROAs that are below C's RC (except, of course, the target IP prefix). The newly issued RCs and ROAs will have the same public keys, child pointers to their subject's directories, and IP prefixes, excluding, of course, the ROA that authorizes the target AS to originate the target IP prefix. But, because they are issued by A, they will have parent pointers to A's RC, and will be placed in A's publication point.

5. **Overwrite the child RC.** Replace B's original RC with the one created in step 1.

At this point, all the original RCs on the path from the revoker's RC and the target IP prefix become invalid, so that the target IP prefix also becomes invalid. Note also that

- The child B gets a new RC; the ROAs of B are covered by B's new RC and remain valid; the same is true for any descendants of B.

- RCs and ROAs of C and below become invalid; however, these RCs and ROAs have been reissued from the revoker's publication point.

- All parties retain the ability to (autonomously) issue RCs and ROAs for the same prefixes as before, except supersets of the target prefix.

The bottom line is that all (IP prefix, origin AS) mappings, apart from the those of the target ROA, remain valid according to the RPKI, and the surgical invalidation succeeds.

Detectability. Note that in the procedure above, the revoker never actually revokes any certificates; instead, it *overwrites* an old RC and issues a few new ROAs and RC. We'd naturally expect to always be able to detect revocations by referring to the CRL; the procedure above suggests that this is not always possible.

However, we note that all the reissued RCs and ROA except the new RC of the child B are now signed by the new issuer and appear in a new directory: namely, the directory belonging to the revoker. Moreover, the RPKI repository now contains pairs of RCs for a single public key published at different publication points (for C and its descendants). Again, both these issues

create an unusual repository state that may be detectable, but does not impact the validity of any ROAs except the target one.

We note that, to reduce detectability, the revoker need not grandparent any RCs; the only impact of doing this is to remove the ability of (some of) his descendants to issue future ROAs themselves, and will not affect the validity of any existing (prefix, ASN) pairs.

We discuss how detectability is reduced in the case the revoker is closer to the target (Sections 2.2.1 and 2.2.2) and how to reduce it in general in Section 2.4 and Section 2.5.

2.4 Reducing detectability by convincing descendants to play along

As pointed out above, the surgical invalidation scenarios may leave the repository with some invalid RCs and ROAs, as well as pairs of RCs that have the same public key but are published at a different point. This can make the revoker's actions easier to detect. In addition, a revoker may have to issue ROAs; if the revoker is an entity that does not normally issue ROAs (such as an RIR), the revocation may be even more easy to detect. However, the revoker can leave the repository in a cleaner state if its descendants can be convinced to play along.

Example: Child revocation. For example, in the child revocation scenario of Section 2.2.2, the revoker B can ask its child F to reissue, on its own, a narrowed-down set of ROAs that exclude the target prefix. In our example, B could ask F to issue a ROA for 12.0.0.0/18 to AS 12 and to delete the 12.0.0.0/17 ROA from F's directory.

Why would F want to satisfy the revoker's request? Because the revoker can simply refuse to reissue the new grandparented ROAs for F's prefixes, which are necessary to protect these prefixes from being affected by the revoker's invalidation (*i.e.,* refuse to perform Step 2 in Section 2.2.2). In that case, ROAs for those prefixes would become invalid, potentially leading to significant damage to F. These ROAs were anyway going to expire at some point, so F already has provisions for reissuing them. Thus, F knows that it has to deal with the problem sooner or later, and is likely to want to want to deal with it immediately.

We note an even simpler course of action for B may be to convince F to do the invalidation on its own, as in Section 2.2.1, which is less detectable. It may actually be in F's interest to cooperate, because in this way F at least retains its ability to issue ROAs for the same set of prefixes as before, and F knows that B can perform the invalidation, anyway, whether F cooperates or not.

Example: Grandchild revocation. In the grandchild revocation scenario of Section 2.2.3, we had revoker issue a new RC for its grandchild F, effectively becoming the grandchild's parent. However, instead, the revoker can ask its child B to issue the new RC for F, and overwrite the old RC for F with a new one RC for F that excludes the target prefix, essentially reducing the situation to that of child revocation (Section 2.2.2), which is less detectable.

Again, B would be interested in satisfying A's request, because if it does not, A can decide not to perform Step 4 of Section 2.2.3 and cause *all* of F's ROAs to become invalid. If F is a customer of B and B wishes to keep F happy, then reissuing F's RC with the reduced range could seem to B like the better option. Moreover, by the same reasoning as above, B may be convinced by A to reissue impacted ROAs, or to even convince F to do so, in order to make sure that no ROAs or RCs move out of their original publication point.

2.5 Some preliminary recommendations for the design of a detector

While more analysis is needed to understand exactly how to design a detector that can distinguish suspicious manipulations of the RPKI from normal RPKI churn, our analysis does suggest that the detecting following types of actions might be useful in the design of such a detector:

1. Detecting when ROAs or RCs are deleted from a publication point and its manifest, or overwritten with different (smaller) sets of resources, bound to AS 0, or invalidated via a CRL.

2. Detecting when a newly issue ROAs causes routes that have been advertised in BGP to become invalid.

3. Detecting when grandparenting occurs, especially if it is accompanied by invalidation of a child RC that contains address space that is a superset of the address space in the grandparented RCs and ROAs.

Careful research is required before such a detector can be developed, especially since many manipulations of the RPKI can look similar to normal RKPI churn; for instance, grandparenting for "legitimate" reasons, as described in [6, Section 1], or make-before-break reissuance of RCs due to key rollovers or transfers of address space, or revocation/expiration of RCs and ROAs due to legitimate terminations of business agreements, *etc.*. This is further complicated by the fact that only limited deployments of RPKI exist today, and so it is difficult to tell what "normal" RPKI churn will look like in the future.

3 Impacting prefix reachability.

Now that we've seen how a revoker can cause a target ROA to become invalid, we next consider how this action impacts prefix reachability. The following point is very important:

> The impact of RPKI manipulations on prefix reachability depends on policies used by relying parties.

Suppose a target ROA in the RPKI becomes invalid. How does this affect the ability of *relying parties* (*i.e.,* parties that consume information from RPKI) to reach the target prefix (*i.e.,* the prefix in the invalid ROA)? The answer is that it depends on how relying parties consume information from RPKI. Each relying party uses its own local policies to decide how to consume RPKI information; this decision is *not* specified or recommended in any RFC or other specification. Therefore, to understand the impact of a invalidated ROA, we shall instead consider some plausible policies for consuming RPKI information that could be implemented by a relying party, and discuss their impact on prefix reachability. We consider three policies:

1. using RPKI information to decide which routes to originate for a customer (Section 3.1)

2. using RPKI information to construct prefix filters for stub customers, (essentially, using RPKI information as IRR instance that is used for prefix filtering) (Section 3.2)

3. using RPKI information for origin authentication (Section 3.3)

For each policy, we shall (1) explain the policies, (2) their ability to blunt attacks on BGP, *i.e.,* how this policies can improve routing security, and (3) how these policies can impact target prefix reachability when the target ROA becomes invalid. We shall see that points (2) and (3) are typically at odds; a policy that effectively blunts routing attacks is more likely to have a significant impact on target prefix reachability when the target ROA becomes invalid.

3.1 Using RPKI to decide which routes to originate for customers.

Here, a network operator that originates a prefix for his customer uses the RPKI to verify the correctness of the IP prefix his customer claims to own. As an example, suppose a network operator in AS 42 has a customer from Org C that claims to own a prefix 52.52.0.0/16; the customer requests AS 42 to originate his prefix for him (so that in BGP, the prefix 52.52.0.0/16 is announced by AS 42). Before the network operator agrees to originate prefix 52.52.0.0/16, he checks to make sure that customer Org C is the true owner of the prefix. In this case, the customer Org C brings his RC to the provider, and if the RC is valid, the provider uses this as evidence that the customer owns the prefix.

Effect on routing security. It is important to note that RC alone is *not* bound to the identity of the customer in any way; this is because the subject name on the RC are a non-descriptive string chosen by the RC's issuer (see RFC 6480 Section 2 for details). Put more plainly, Org C's name will not appear anywhere on his RC. Thus, a malicious Org C could exploit this by bringing his provider the RC belonging to some other organization Org V; if the provider decides to originate the prefix in Org V's RC without Org V's permission, the malicious customer has effectively convinced his provider to perform a prefix hijack on Org V. Here are two ways to remedy this vulnerability:

- To deal with the lack of human-identifiable subject names in RCs, the RPKI uses a *Ghostbusters record*, an object listing Org C's RPKI maintainer's name and contact information and signed by Org C's RC (see RFC 6493 for details). Thus, if the RC has a valid Ghostbuster record mapping to the customer, then the provider can verify that the RC really belongs to his customer.

- Another way to validate that a customer really owns a prefix is to request the customer to issue a ROA for that prefix. More specifically, in our setting, the provider AS 42 would require the customer that holds the RC for prefix 52.52.0.0/16 to either (a) issue a ROA binding AS 42 to prefix 52.52.0.0/16, or (b) give the provider access to the secret key corresponding the public key in the customer's RC, and have the provider use that secret key to issue a ROA binding AS 42 to prefix 52.52.0.0/16.

  If the resulting ROA is valid, the provider no longer needs to trust the customer to bring the correct RC; instead, he is presented with cryptographic proof that the customer owns the secret key that corresponds to the public key that is in the RC, which should only be possible if the customer is the true owner of the RC, and therefore also the prefix certified by the RC.

In both cases, this policy can be used as proof that the customer really owns the prefix, and therefore can prevent an inadvertent prefix hijack on the part of the provider.

Impact on prefix reachability. If the customer's RC becomes invalid prior to the customer's request to originate a prefix, the provider might refuse to originate that prefix. However, as origination of a new prefix is done "manually", the provider can also manually verify whether or

not the customer should be allowed to originate the prefix, using data sources other than the RPKI. Moreover, the customer could always try to find another provider that is willing to originate the prefix for him, but this could prove difficult if his RC is invalid. However, it is important to note that if *no* provider is willing to originate a prefix with an invalid RC, then the prefix then becomes unreachable *to every AS in the Internet*, even those that do not rely on information from the RPKI.

3.2 Using RPKI to construct prefix filters for stub customers

A provider uses RPKI information to create a whitelist of prefixes his stub customer should be allowed to originate, and drops all announcements made by this stub customer for prefixes not on this whitelist. This is the same functionality provided today by prefix filters with the exception that in the subsequent discussion we will only consider filtering for *stub* customers, (*i.e.,* customers with no customer's of their own), rather than the filtering the entire customer cone, *i.e.,* customers of customers, customers of customer's customers, *etc.*.

Implementation. This policy can be implemented automatically using a prefix list construction tool like IRRpowerTools [3] that downloads information from an Internet Routing Registry (IRR) instance generated from RPKI repositories, (but does not use information from any other IRR instance). For example, RIPE has an prototype IRR instance that serves RPSL objects created from validated ROAs, see [24]. Note also the IRRpowerTools can automatically construct prefix whitelists, updating these regularly (*e.g.,* once per day) based on the changing information in the IRR.

Effect on routing security. If *every* provider of a given stub customer uses this policy, this policy can prevent prefix hijacks by that stub customer.

Effect on prefix reachability. Suppose a target stub's ROA becomes invalid. The following conditions must hold for target prefix to become unreachable *for every AS in the Internet*, even those that do not consume RPKI information:

1. every provider of the target stub uses the policy described in this section (*i.e.,* construct prefix filters for stub customers using an IRR instance generate from valid ROAs), and

2. a route object (that maps a prefix to an origin AS) is removed from the IRR instance when its corresponding ROA becomes invalid.

Since this policy only applies to a provider's stub customer, a non-stub's prefix cannot become unreachable as a result of this policy. [4]

3.3 Use RPKI information for origin authentication.

Unlike the previous policy where RPKI information only impacts routes originated by a provider's stub customer, with full origin authentication the RPKI can impact routes originated by *any* AS. For details on how origin authentication is implemented, see RFC 6483 and BGP Prefix Origin Validation Draft [23]; for our purposes, it suffices to note that BGP routes seen at a relying party's router will be automatically classified as having one of the following *validation states*:

[4] One can imagine some hybrid of this policy where "as-sets" from the IRRs are used alongside with ROAs to construct prefix filters for ASes deeper in the customer cone (*i.e.,* customers of customers, customers of customers of customers *etc.*). Its difficult to assess the security of doing this, since "as-sets" are not certified by the RPKI. However, if this policy is adopted, then the risk above could extend to non-stub customers as well.

"valid", "invalid", or "not-found" (synonymously, "unknown") based on ROA information from the RPKI. It is then up to the relying party's local policy to use these validation states as it sees fit — that is, the route's validation states can be ignored, or used to raise alarms, or to impact route selection, *etc.*. Here we focus on the case where validation state is used to impact route selection. BGP Prefix Origin Validation Draft Section 5 states:

> Policies which could be implemented include filtering routes based on validation state (for example, rejecting all "invalid" routes) or adjusting a route's degree of preference in the selection algorithm based on its validation state. The latter could be accomplished by adjusting the value of such attributes as LOCAL_PREF. Considering invalid routes for BGP decision process is a pure local policy matter and should be done with utmost care.

The RFC does *not* specify how validation state should impact route selection policies. Since there is a myriad of such policies, we will consider only two plausible policies here: (1) depreference invalid (Section 3.3.1), and (2) drop invalid (Section 3.3.2). Note that both of these policies are mentioned in the quote above.

3.3.1 Depreference invalid.

The policy. Reduce the "degree of preference" for an invalid route. The terminology "degree of preference" is used because the impact of a route's validation state can occur anywhere in the multi-step BGP decision process [8]. There are multiple ways the degree of preference could be selected. Here are a few possibilities:

1. **Strictly prefer valid.** Prefer any valid route over any invalid route. (This is the strictest depreference invalid policy.)
2. **Prefer valid after LOCAL_PREF.** A more complex policy could first rank routes by LOCAL_PREF and then break ties on validation state. This policy, for example, could be used to realize a preference for an *invalid customer route* over a *valid provider route*, while preferring a *valid customer route* over an *invalid customer route*.
3. **Prefer valid after AS_PATH length.** Another possible policy might be to prefer a valid route over an invalid route only if both routes have equal LOCAL_PREF (where the LOCAL_PREF function ignores validation state) and AS-path length.

Remember that longest-prefix match forwarding only allows for comparing the preferences of routes with *exactly matching* prefixes. Thus, the key point to keep in mind is that if a route is invalid for a given prefix, and there is a valid route with a higher degree of preference (according the relying party's routing policy) for the *exact same prefix*, than than the valid route is chosen for routing. However, if no such valid route is available, then the invalid route is chosen.

Impact on routing security. This policy **cannot** prevent subprefix hijacks. To see why, consider the following example paraphrased from RPKI-Based Origin Validation Operation Draft [7, Section 5]. Suppose that there is a valid ROA for prefix 10.0.0.0/16 with origin AS 42 and maxlen 24, and suppose that AS 42 sends a BGP announcement for prefix 10.0.0.0/16 *only*. Suppose a subprefix hijacker (AS 666) announces a BGP route for prefix 10.0.0.0/24 (with origin AS 666).

Even though this route is invalid, this is the only route that is announced in BGP for prefix 10.0.0.0/24. Since there is no valid route for 10.0.0.0/24, longest prefix-match forwarding will choose the invalid route, and traffic for address 10.0.0.0/24 will be sent to the hijacker. Note that the Pakistan Telecom/YouTube incident was a subprefix hijack of this type [26].

This policy can sometimes blunt the impact *prefix hijacks*, *i.e.,* where the hijacker announces a prefix of the *exact same length* as the victim prefix. (That is, in the example above, the hijacker would originate 10.0.0.0/16 instead of 10.0.0.0/24.) However, the degree to which this attacks are prevented depends on the degree of preference for valid routes uses by the relying parties. For instance, if *all* AS in the Internet use the "Strictly prefer valid" policy (item 1 above), then all prefix hijacks are prevented. However, if some ASes use more complex "depreference invalid" policies, then the hijacker's path might still be preferred over the legitimate path. For example, if all ASes use the "Prefer valid after AS_PATH length" policy above, these ASes would choose the hijacker's route if the hijacker's path is of equal LOCAL_PREF but is shorter than the legitimate path. Finally, we note that one AS's policies can influence the routes selected by another AS, even if that AS uses the "Strictly prefer valid" policy. To see how, suppose an AS implements the "Prefer valid after AS_PATH length" policy above, and the hijacker's path has equal LOCAL_PREF and is shorter than the legitimate route. In this case, that AS would select and propagate the hijacked route to its neighbor, so that the invalid route is *the only route* learned by this neighbor; thus this neighbor is forced to select the invalid route, even if it uses the "Strictly prefer valid" policy.

Impact on prefix reachability. Suppose a target ROA becomes invalid. The route announced for the target prefix by the AS in the target ROA also becomes invalid. We now have two cases:

- If there is no alternate route announced in BGP with a valid origin the target prefix, there is no impact on prefix reachability; traffic still goes to the (invalidated) target prefix.
- If there is alternate route with a valid origin announced in BGP for the target prefix, this alternate valid origin can impact prefix reachability, potentially attract convincing some ASes to route to this alternate route. (How many ASes would route to the alternate route depends on the topology of the network and type of depreference invalid policies used by other ASes in the network.)

Note. The complexity of the depreference invalid policy (*i.e.,* the fact that many such policies exist, depending where validity state is placed in the BGP decision process, AND the fact that different ASes could place validity state at *different* points in the BGP path selection process) suggest that a more detailed analysis of the impact of this policy may be needed. We leave this analysis to future work.

3.3.2 Drop invalid

The policy. This (strict) policy drops any invalid BGP route. That, a relying party will never send traffic along an invalid route.

Impact on routing policy. This policy stops all prefix hijacks, and all subprefix hijacks.

Impact on prefix reachability. If a target ROA becomes invalid, then the target prefix is no longer reachable at the router that is dropping invalid routes.

Acknowledgments

We thank Tony Tauber for many useful discussions. We also thank the members of the FCC CSRIC Working Group 6 on Secure BGP Deployment for inspiring this study.

References

[1] The open network initative. http://opennet.net/.

[2] rcynic software. http://trac.rpki.net.

[3] IRR power tools. http://sourceforge.net/projects/irrpt/, 2011.

[4] S. Amante. Risks associated with resource certification systems for internet numbers, 2012.

[5] R. Austein, G. Huston, S. Kent, and M. Lepinski. *RFC 6486: Manifests for the Resource Public Key Infrastructure (RPKI)*. Internet Engineering Task Force (IETF), 2012. http://tools.ietf.org/html/rfc6486.

[6] R. Bush. *Responsible Grandparenting in the RPKI*. Internet Engineering Task Force Network Working Group, 2012. http://tools.ietf.org/html/draft-ymbk-rpki-grandparenting-02.

[7] R. Bush. *RPKI-Based Origin Validation Operation*. Internet Engineering Task Force Network Working Group, 2012. http://tools.ietf.org/html/draft-ietf-sidr-origin-ops-19.

[8] Cisco. Bgp best path selection algorithm: How the best path algorithm works. Document ID: 13753, May 2012. http://www.cisco.com/en/US/tech/tk365/technologies_tech_note09186a0080094431.shtml#bestpath.

[9] J. Cowie. Rensys blog: China's 18-minute mystery. http://www.renesys.com/blog/2010/11/chinas-18-minute-mystery.shtml.

[10] J. Cowie. Wikileaks: Moving target. Renesys blog, December 7 2010. http://www.renesys.com/blog/2010/12/wikileaks-moving-target.shtml/.

[11] J. Fallows. The connection has been reset. *The Atlantic*, March 2008.

[12] G. Huston. *RFC 6485: The Profile for Algorithms and Key Sizes for Use in the Resource Public Key Infrastructure (RPKI)*. Internet Engineering Task Force (IETF), 2012. http://tools.ietf.org/html/rfc6485.

[13] G. Huston, R. Loomans, and G. Michaelson. *RFC 6481: A Profile for Resource Certificate Repository Structure*. Internet Engineering Task Force (IETF), 2012. http://tools.ietf.org/html/rfc6481.

[14] G. Huston, R. Loomans, and G. Michaelson. *RFC 6487: A Profile for X.509 PKIX Resource Certificates*. Internet Engineering Task Force (IETF), 2012. http://tools.ietf.org/html/rfc6487.

[15] G. Huston and G. Michaelson. *RFC 6483: Validation of Route Origination Using the Resource Certificate Public Key Infrastructure (PKI) and Route Origin Authorizations (ROAs)*. Internet Engineering Task Force (IETF), 2012. http://tools.ietf.org/html/rfc6483.

[16] G. Huston, G. Michaelson, and S. Kent. *RFC 6489: Certification Authority (CA) Key Rollover in the Resource Public Key Infrastructure (RPKI)*. Internet Engineering Task Force (IETF), 2012. http://tools.ietf.org/html/rfc6489.

[17] S. Kent, D. Kong, K. Seo, and R. Watro. *RFC 6484: Certificate Policy (CP) for the Resource Public Key Infrastructure (RPKI)*. Internet Engineering Task Force (IETF), 2012. http://tools.ietf.org/html/rfc6484.

[18] M. Lepinski, editor. *BGPSEC Protocol Specification*. IETF Network Working Group, Internet-Draft, July 2012. Available from http://tools.ietf.org/html/draft-ietf-sidr-bgpsec-protocol-04.

[19] M. Lepinski, A. Chi, and S. Kent. *RFC 6488: Signed Object Template for the Resource Public Key Infrastructure (RPKI)*. Internet Engineering Task Force (IETF), 2012. http://tools.ietf.org/html/rfc6488.

[20] M. Lepinski and S. Kent. *RFC 6480: An Infrastructure to Support Secure Internet Routing*. Internet Engineering Task Force (IETF), 2012. http://tools.ietf.org/html/rfc6480.

[21] M. Lepinski, S. Kent, and D. Kong. *RFC 6482: A Profile for Route Origin Authorizations (ROAs)*. Internet Engineering Task Force (IETF), 2012. http://tools.ietf.org/html/rfc6482.

[22] S. Misel. "Wow, AS7007!". Merit NANOG Archive, apr 1997. http://www.merit.edu/mail.archives/nanog/1997-04/msg00340.html.

[23] P. Mohapatra, J. Scudder, D. Ward, R. Bush, and R. Austein. *BGP Prefix Origin Validation*. Internet Engineering Task Force Network Working Group, 2012. http://tools.ietf.org/html/draft-ietf-sidr-pfx-validate-09.

[24] P. Palse. Serving roas as rpsl route[6] objects from the ripe database. RIPE Labs, June 2010. https://labs.ripe.net/Members/Paul_P_/content-serving-roas-rpsl-route-objects.

[25] I. G. Project. In important case, RIPE-NCC seeks legal clarity on how it responds to foreign court orders, 2011. http://www.internetgovernance.org/2011/11/23/in-important-case-ripe-ncc-seeks-legal-clarity-on-how-it-responds-to-foreign-court-orders/.

[26] Rensys Blog. Pakistan hijacks YouTube. http://www.renesys.com/blog/2008/02/pakistan_hijacks_youtube_1.shtml.

[27] L. S. Smith. *H.R.3261 – Stop Online Piracy Act*. October 26, 2011.

[28] M. Wählisch, O. Maennel, and T. Schmidt. Towards detecting BGP route hijacking using the RPKI. In *Poster: SIGCOMM'12*, pages 103–104. ACM, 2012.