The ISP Column
A monthly column on things Internet

April 2020
Geoff Huston

RPKI and Trust Anchors

I've been asked a number of times: "Why are we using as distributed trust framework where each of the RIRs are publishing a trust anchor that claims the entire Internet number space?" I suspect that the question will arise again the future so it may be useful to record the design considerations here in the hope that this may be useful to those who stumble upon the same question in the future.

Trust anchors are what relying parties ("relying parties" are those folk who want to use a PKI to validate digitally signed attestations) hold in order to validate all digitally signed artefacts in that PKI. Validation in the X.509 certificate world requires that the relying party construct a chain of certificates where each link in the chain corresponds to a certification authority whose private key has signed the next (or immediate subordinate) public key certificate in the chain.

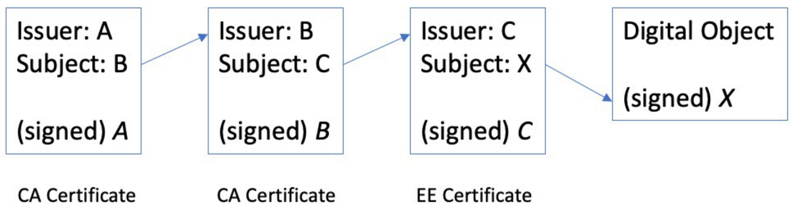

Figure 1 – X.509 Certificate Chain

This chain of issuer/subject relationships ends with the End Entity Certificate of the public key being used in the digital certificate. At the other end of this chain is a self-signed certificate that the relying party is prepared to trust under all circumstances.

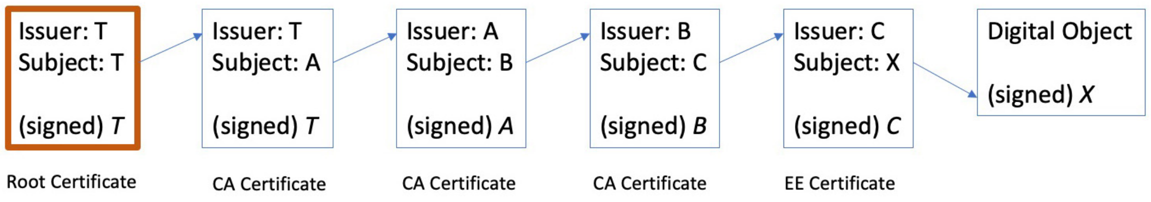

Figure 2 – X.509 Certificate Chain with Trust Anchor

Normally, within the context of particular PKI the trust anchor(s) would be widely distributed. Each relying party is expected to learn these trust anchors in a manner that they are prepared to trust. The reasons for this are hopefully pretty obvious, but to illustrate what can go wrong if a relying party just believes anything they are told, then think about the following: A would-be attacker could simply represent a self-signed certificate that they have created for this attack to be a trust anchor and present the intended victim with a digitally signed object, a chain of certificates and this purported trust anchor.

Conventionally, a trust anchor within a PKI is widely distributed and locally cached by every relying party within this PKI. The implication is that its strongly preferred that the trust anchor of a PKI to be highly stable and change infrequently, if at all, as any changes have to be comprehensively propagated across all relying parties. This is analogous to the role of the Key Signing Key (KSK) in DNSSEC. As we've seen in the recent exercise to change the KSK value, making sure that every relying party is in sync with this change and replaces their trust in the old trust anchor with trust in the new trust anchor is a tricky affair. Therefore, we would like trust anchors to be stable and long lived, and conventionally we would change the trust anchor key value infrequently, if at all. And if it is to change it would be better to have this trust anchor change at predicted and well-signalled times so that relying parties can manage their trust in sync with these changes.

Now let's add one further consideration from the Resource PKI (RPKI). The trust anchor needs to also include a set of IP number resources that are within the 'scope' of this trust anchor. The implication is that the trust material needs to be changed whenever the associated set of "in scope" number resources change. This is true for self-signed trust anchor certificates as much as it is true for all other RPKI certificates. While changes in the key values can be planned in advance, changes in resource holding are not necessarily so predictable, which has design implications for the trust anchors of the RPKI.

The Resource PKI introduced a subtle twist to the conventional interpretation of X.509 Public Key certificates.

Informally, a PKI constructs a structure of transitive trust that allows a relying party to answer the question: Is this digital signature *genuine*? This question is transformed into a slightly different question: Is the public key part of the key pair used to generate the signature one that belongs to the party who is claiming to have generated this signature? Given that the party asking that question may not know the signing party or their public key, the PKI is useful to answer a variant of this question: Are there people I trust who either know the signing party and can assure me that the key pair belongs to the party who is claiming to have generated this signature or themselves trust others who can provide this assurance? The essential point here is that the conventional role of the PKI is all about keys and identity. "Is this your key?" is the point of the PKI.

The RPKI is different. It's not about assertions of identity. It's about assertions of ownership. RPKI X.509 certificates include a set of IP number resources (IP addresses and/or Autonomous System Numbers). The question the RPKI is intending to answer is "Is this your address?" This is no longer about the association of keys with identity but keys with control over IP resources.

There is a need to balance conflicting goals. In theory we want long-lived stable trust anchors, just like the KSK for DNSSEC. The issue is that if we change a trust anchor, then we need every relying party to delete their old trust anchor(s) and load new ones. Some will, but just like our experience of the KSK roll, some won't. And as the RPKI matures and the implementations of relying party tools diversifies it is hopelessly naive to think that every implementation will treat the RPKI trust anchors as highly volatile and will continually check for a change in the trust anchor. It's inevitable that some implementations will treat the current trust anchor as a static value. On the other hand, if these relying parties treat the trust anchor as volatile, then they will need to continually check with the original trust anchor publication point to check for any updates in this material. This makes the trust anchor publication points a critical resource. A DDOS attack on a trust anchor publication point would pose a risk to the entire RPKI as these constantly probing relying parties would need to make things up because they could not reach the trust anchor publication point. Its highly preferable to use trust anchors that have highly stable material.

The source of 'truth' for the RPKI is the collection of Internet registry data. When an Internet Registry allocates a number block to a recipient subordinate registry not only is this transaction recorded in the registry, but when the address recipient passes a certificate signing request to the parent registry then the registry would issue a certificate to bind the allocated address block to the public key of the recipient. This is intended to hold for all parts of the RPKI from the Root Certificate to the End Entity certificates at the leaf points of the hierarchy.

The implication of this model of operation is that the RPKI is intended to precisely follow the address allocation actions of Internet registries. If this is all there is to it then the story would stop here. However, in response to the exhaustion of the fee pool of IPv4 addresses the regional address policy communities adopted the concept of address transfers, both for intra-region and inter-region transactions. While the RPKI was designed to describe the hierarchical allocation of addresses in certificates, the 'horizontal' movement of IP addresses between registries presented some quite fundemantal issues to the design of the RPKI.

To look at the implications of transfers on the RPKI structure, let's look at an inter-RIR transfer of an address from the perspective of the RPKI.

A entity serviced by the RIR X, ISP A, transfers an address prefix to ISP B, who is an entity serviced by the RIR Y. RIR X will need to revoke its certificate it had issued to ISP A, and if ISP A still holds other number resources it will need to issue a new certificate for ISP A with a reduced set of number resources. RIR Y will need to issue (or re-issue) its certificate for ISP B, with the newly issued certificate including the transferred address prefix.

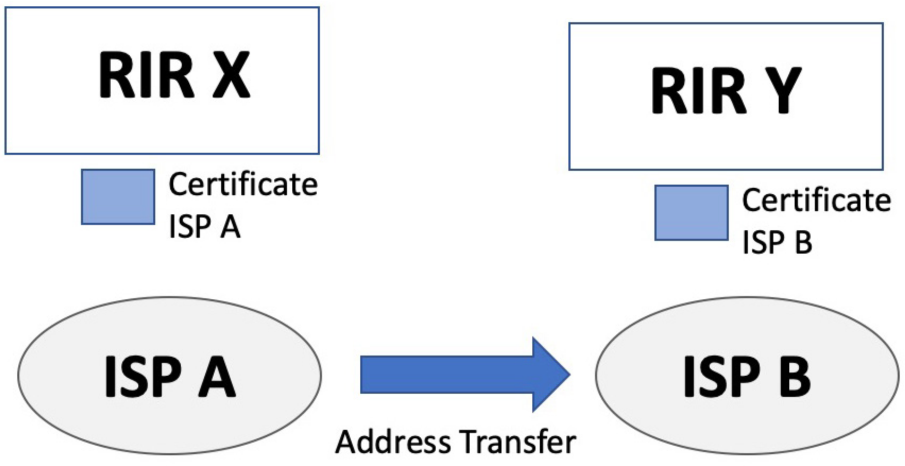

Figure 3 – Address Transfer Scenario

The choice of trust anchor models affects the complexity of this set of certificate actions.

If each of these RIRs publish a trust anchor that includes all resources (a "0/0" self-signed certificate) then the actions involved in the transfer are quite straightforward:

1. RIR X issues a new certificate for ISP A that does not contain the transferred resources
2. RIR X revokes the old certificate for ISP A
3. RIR Y issues a new certificate for ISP B that contains the transferred resources
4. RIR Y revokes the old certificate for ISP B

The consideration of "liveness" determines the order of these actions. If you want to allow the network using this address to always be "covered" by an RPKI certificate that can be validated at all times, then

RIR Y would issue a new certificate first, and the revocation of the old certificate would be the final act of the transfer (i.e., in sequence the actions would be 3,1,4,2).

What if each RIR publishes a trust anchor that does not list “0/0” but instead lists precisely those addresses that are listed in their local registry and no more. Now the sequence is a little more involved the transfer involves a change in the trust anchors of the RIRs involved in the transfer:

1. RIR Y issues a new trust anchor that includes the to-be-transferred resources
2. RIR Y issues a new certificate for ISP B that contains the to-be transferred resources
3. RIR X issues a new certificate for ISP A that does not contain the transferred resources
4. RIR Y revokes the old certificate for ISP B
5. RIR X revokes the old certificate for ISP A
6. RIR X issues a new trust anchor that does not include the transferred resources

The above process is fragile in a number of ways. The actions of the RIRs are in a particular order, but the actions of relying parties are not. What if a relying party does not see the changed RIR Y trust anchor, but picks up the new certificate for ISP B first? From the perspective of the relying party that certificate is invalid because it is not ‘covered’ in the RIR Y’s trust anchor. To make this process more robust you need to introduce delays to allow relying parties to keep up, and these delays should be measured in days rather than hours. A modified process is:

1. RIR Y issues a new trust anchor that includes the to-be-transferred resources
2. wait
3. RIR Y issues a new certificate for ISP B that contains the to-be transferred resources
4. RIR X issues a new certificate for ISP A that does not contain the transferred resources
5. RIR Y revokes the old certificate for ISP B
6. wait
7. RIR X revokes the old certificate for ISP A
8. RIR X issues a new trust anchor that does not include the transferred resources

There are a lot of transfers happening. And there is little doubt that there will be a higher intensity of transfers in the future, so this 8-step process may be executed many times in parallel. This implies that the trust anchors for the RIRs would be in a constant state of flux and relying parties would have to continually refer back to these trust anchor publication points to assure themselves that their locally cached copy of the trust anchor is current.

The alternative is that each RIR uses a trust anchor that contains a 0/0 resource set. This way changes to the trust anchors would be limited to changes in the key material, and this can be managed in a far more controlled fashion.

The conclusion is that if the RIRs each publish a trust anchor, then the use of a 0/0 resource set in these trust anchors allows for stability in this material so that relying parties do not have to constantly re-check the state of their root of trust. Other approaches to an RIR-based trust anchor set are more fragile.

The alternative course of action is that the RIRs do not publish their own trust anchors. This alternative path envisages the IANA publishing a single trust anchor for the entire RPKI. This approach was the starting position in the design of the RPKI.

There is a lot to be said for an IANA-signed 0/0 trust anchor. It’s stable, long lived and can be securely managed. All of these are desirable attributes of a trust anchor.

However, the question arises: What would be in the certificates IANA issues for each RIR?

The conventional answer is the same answer that the RIR’s use in the RPKI, namely that the RPKI is an exact mirror of current registry contents. An RPKI certificate is not just invented but is based on the registry contents. Accordingly, the IANA-issued certificates would be based on the information in the IANA registry (https://www.iana.org/numbers) regarding the resources allocated to each RIR to manage.

In this context let's look once more at this transfer of resources from ISP A to ISP B. Is the IANA CA involved in this transfer?

One approach is that IANA certificates that certify RIR X and RIR Y need to be altered to reflect this transfer, shifting the resource across from the certificate for RIR X to RIR Y (Figure 4).

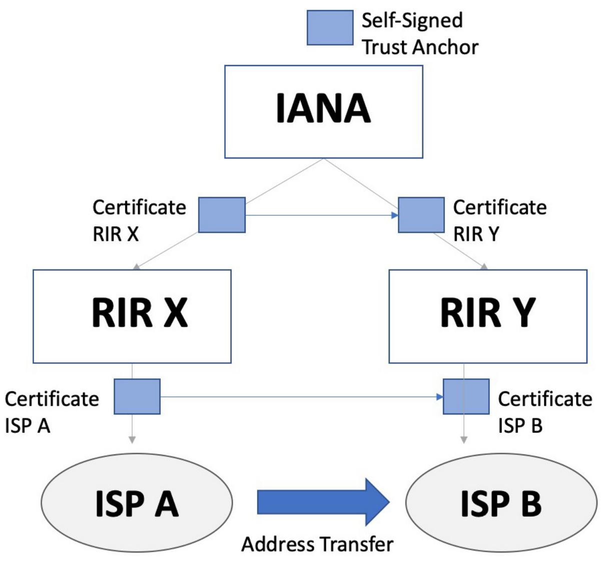

Figure 4 – Address Transfer with IANA Certificates

This transfer is not in the IANA number registry, as the IANA registry only records allocations from the IANA to RIRs and reservation actions by the IETF. This implies that this approach would see the IANA issuing a certificate where the resource contents contradicts the IANA registry. We could fix this up by instituting a new operational process where all inter-RIR transfers are processed by the IANA and the IANA registries are updated to reflect the current disposition of all transferred resources. This proposal opens up a set of policy questions as it places the IANA in the position of "approving" each and every address transfer and entering this into the IANA registry. It also raises the question: "What is the IANA registry?" It would no longer represent a trusted and accurate log of IANA's own allocation actions in the past, but a compendium of transactions that other parties have told the IANA. Presumably the registries would devise a secure and authenticated manner of informing IANA in a way that cannot be repudiated, and that is provably genuine on both sides of the transfer and these tests need to be verifiable by anyone who cares to look, but the basic observation remains that IANA is no longer recording its own actions but is acting as a registry of actions taken by other registries.

My own reaction to this possible model is that what may be most difficult issue for the RIRs is not the operational considerations, challenging as they might be to execute such transactions reliably every time, but the policy question. It is difficult to understand how the RIR communities would accept placing the IANA in a role that essentially oversees and tacitly is required to stamp its imprimatur by approving each and every micro-action that is an address transfer. What happens if the IANA ever disapproves and refuses to process an address transaction proposed by two RIRs?

If this is not an acceptable arrangement, then the consequent question is: "How can we remove IANA from the loop?" The alternative is that IANA issues certificates to each RIR that is an exact replica of the historical allocation actions described in the existing IANA registry. When ISPs A and B conduct their transfer, then IANA is not involved and IANA certificates for RIRs X and Y cannot change. IANA did not conduct the transfer so IANA has no grounds to make changes to its registry. Under these constraints how can the transferred resources be certified? The only path forward is for RIR X to certify RIR Y for the resources, and RIR Y to certify ISP B using this “cross-RIR” certificate as its “parent”. Now ISP B might end up with 2 certificates: one for resources that were allocated IANA ⇒ RIR Y ⇒ ISP B and a second for resources IANA ⇒ RIR X ⇒ RIR Y ⇒ ISP B. These two certificates cannot be merged.

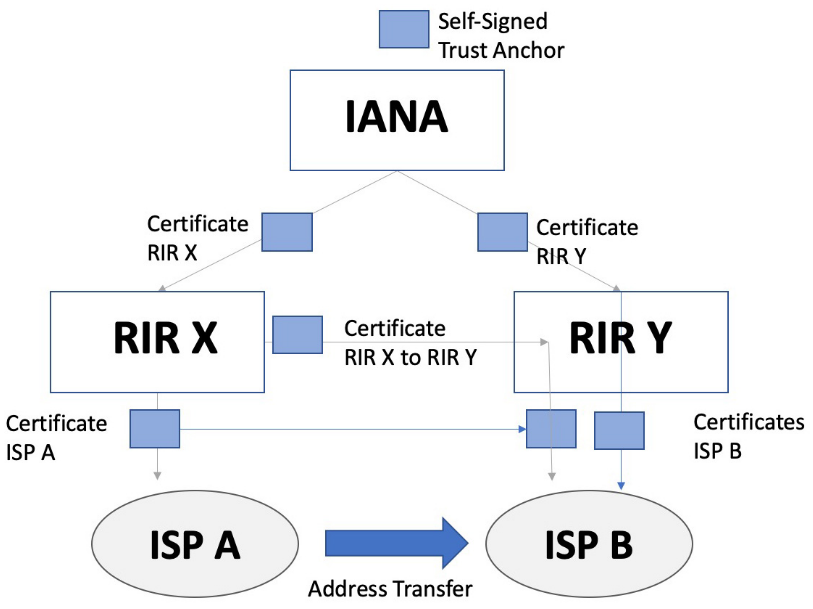

Figure 5 – Transfer with Cross-RIR Certificates

What happens if ISP B subsequently transfers all of its resources to ISP C, an entity serviced by RIR Z? Its none of RIR X’s business anymore and it really should not be placed in the position of having to ‘approve’ this transaction given that it has no relationship with either party to this subsequent transaction.

If we just want to conduct this second transaction as being between RIR Y and RIR Z then ISP C will be the subject of a certificate whose validation path is IANA ⇒ RIR X ⇒ RIR Y ⇒ RIR Z ⇒ ISP C and also be the subject of a second certificate whose validation path is IANA ⇒ RIR Y ⇒ RIR Z ⇒ ISP C. Again, these two certificates cannot be merged. As more transfer take place, the certificate structure takes on increasing complexity. The result is that the alternative to an IANA trust anchor being included in every micro-transfer is a situation where the RPKI certificate system becomes extraordinarily complex very quickly, and resource holders may hold a large collection of certificates to describe their address holding, even when the holder has registered all their addresses with a single RIR.

It’s not so bad is it? As long as those certificates can be validated and don’t contradict each other then all would be good, right? At least from a technical point of view, never mind the possible the operational burden, it’s all good isn't it? What’s the problem here?

What we wanted was a system that augmented the registry with digital keys. Possession of a private key allowed a resource holder to say: “That’s my resource and my RIR will validate my digital signature if I sign a digital attestation to that effect”. It’s a “strong” version of the *whois* registry report tool. What we didn't want was a X.509 public key certificate system that mandated changes to the RIR structure or changes to the IANA/RIR model. To achieve that very simple outcome and not buy into a whole set of externalities that introduce complexity and fragility, or introduce changes in the policy and organisational landscape between the RIRs and between the RIRs and the IANA then the decision space as to how to design trust in the RPKI is a very constrained space.

It we want to avoid highly volatile trust anchors and want to avoid blossoming complexity in certificate issuance and management, and avoid rewriting the policy framework of the relationship between the

RIRs and the IANA, then all that's left is the option to use a trust anchor for each RIR to issue a self-signed trust anchor with a “0/0” resource set. Every other conventional model creates additional complexity and brittleness, and some models require a re-working of the IANA/RIR roles and framework, a task that nobody appears keen to even contemplate!

Of course, there is always the possibility to head into other more unconventional approaches.

One approach is to discard the concept of 5 (or 6) distinct issuers and enrol all these entities as Registration Agents of a single resource CA. This way all issued RPKI certificates from the RIRs (and IANA potentially) would be issued from a single CA, and every resource holder could have all their resource holdings certified in a single certificate irrespective of transfers. I suspect that the policy and operational issues in setting up this shared structure and potentially setting up a single shared number registry to audit potential collisions and gaps in the registry far outweigh the policy and practical issues of using certificate registration agent roles, but both aspects of such an arrangement face the barrier of posing a major change to the current arrangements for the number registry function.

Another approach is to observe that the RPKI is never queried as an identity or role verification. We use other PKIs for that. The RPKI can verify attestations about a number resource with a key. It is overkill to perform validation of all resources in an RFC3779 extension when in fact the relying party’s query is about the validation of a particular number resource. The question that the validation process is implicitly asking is “did you, the CA, actually issue this number resource to the entity who holds public key?” At this point the element of X.509 heresy enters the room. A holder of a set of resources which have been allocated through different allocation paths as a result of transfers across registries could submit the same certificate signing request to multiple CAs. If a CA signs the request and issues a certificate it is not attesting that it necessarily allocated all of the resources in the certificate, but it did allocate some of them. This is intended to allow merging of certificates and would be useful in cases such as show in Figure 5. This approach is a hybrid of the strict hierarchy of the RPKI as it is currently defined and a web of trust model where a key may be certified by multiple CAs. Obviously this is complete X.509 heresy, but if you are curious about how this approach could be used in the content of the RPKI and would like to understand this further, then there is a presentation pack that goes into this bad idea in gory detail (https://www.potaroo.net/presentations/2014-03-04-draft-sidr-validation.pdf). Don’t try this at home!

But in the same way that the RPKI was not intending to re-phrase the relationship between the RIRs and IANA, the RPKI was also not intending to invent its own unique public key infrastructure, so both of these alternatives remain as though exercises with little practical relevance to the design of the RPKI trust anchor.

Why the RIRs use a trust anchor set of five RIR-based 0/0 self-signed certificates?

In many ways the answer is a forced one, as there is no real choice between multiple viable alternative designs. There is no other feasible design for the root of trust in the RPKI given the constraints of both the X.509 certificate structure, the demands placed on the system by the introduction of resource transfers between registries, and the constraints of the organisational landscape in which the resource management system operates.

Disclaimer

Author

Geoff Huston AM, B.Sc., M.Sc., is the Chief Scientist at APNIC, the Regional Internet Registry serving the Asia Pacific region.

www.potaroo.net