RPKI Monitor: Modeling & Measuring INR conflicts in RPKI

Xiang Liu[1], Min Li[1], Yu Zhang[1], Di Ma[2]

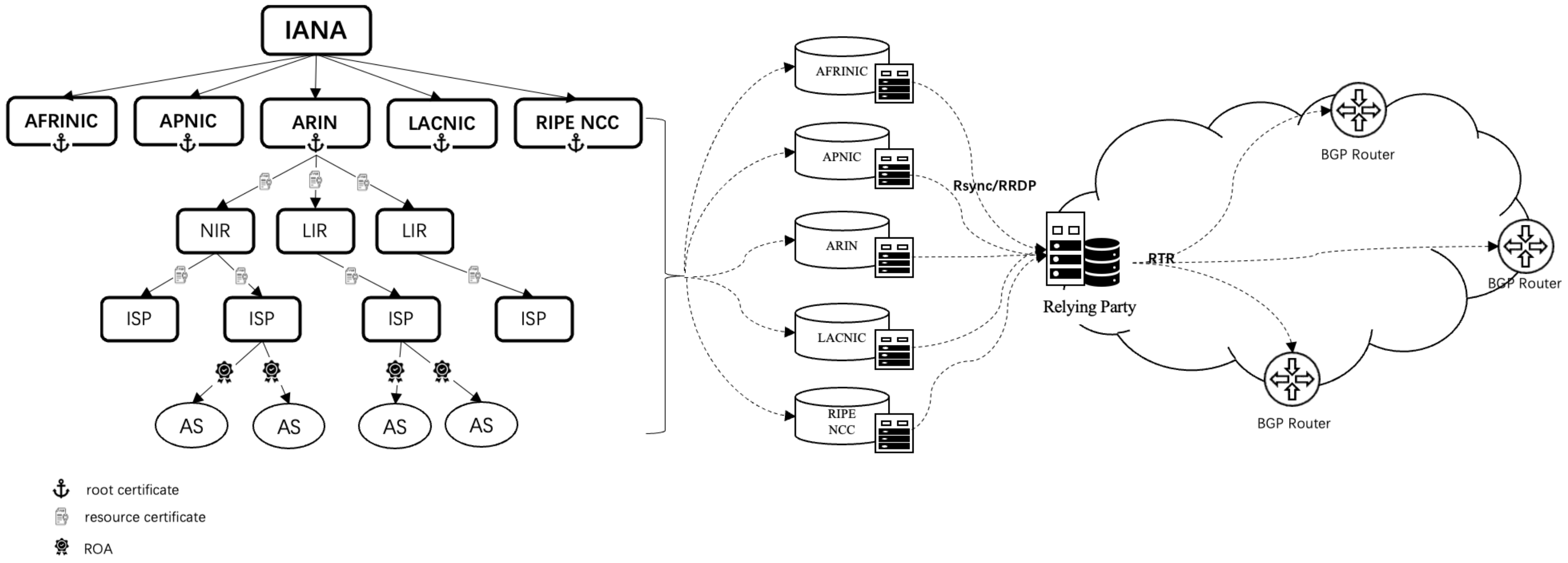
[1]Pengcheng Laboratory, [2]ZDNS

February 2025, Malaysia

Outline

Briefly recap the RPKI system

- **Centralized Hierarchical** trust model: anchored in RIRs
- **Distributed** repositories
- Hosted Model v.s. Delegated Model



Reasons to cause INR conflicts

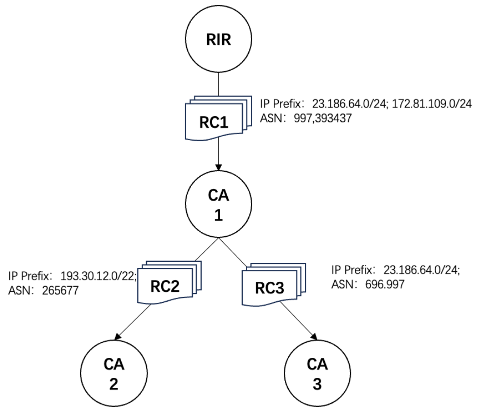
- **Overlapping INR delegation of RIRs**: Number Resource Organization (NRO) recommended expanding RPKI trust anchor's certification scope to all IP addresses and AS numbers.
- **Mis-behavior authorities**: RPKI's hierarchical trust model allows authorities to perform unilateral operations on child CA.
- **AS 0**: Conflicts may arise when an authority authorizes the same IP prefix to multiple ROAs with different ASNs, including AS0.
- **INR transfer**: Though the RPKI eco-system follows the "make before break" principle, and the INR conflicts caused by INR transfer can be regarded as legal temporally, it is apparently unacceptable if those conflicts exist for a long time.

INR Conflicts Model

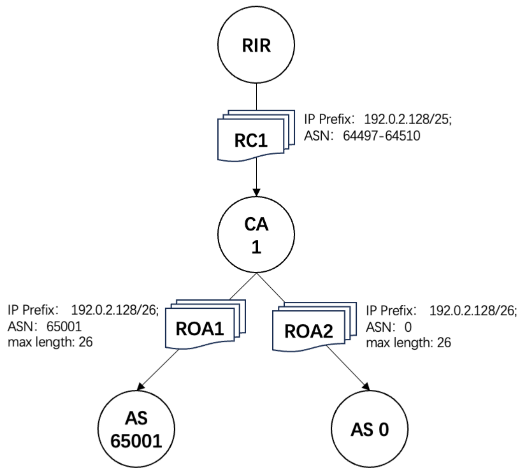
Generally there are three kinds of INR conflicts

- Illegal delegation (including allocation and authorization)
- repetition
- overlapping

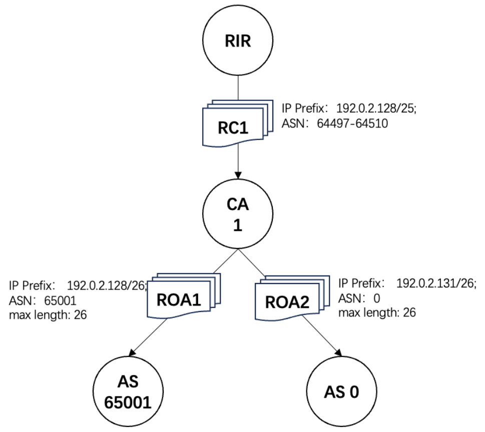
Type A: Illegal INR delegation


RIR
RC1
IP Prefix: 23.186.64.0/24; 172.81.109.0/24
ASN: 997,393437
CA 1
IP Prefix: 193.30.12.0/22;
ASN: 265677
RC2
RC3
IP Prefix: 23.186.64.0/24;
ASN: 696.997
CA 2
CA 3

Type B: INR Repetition



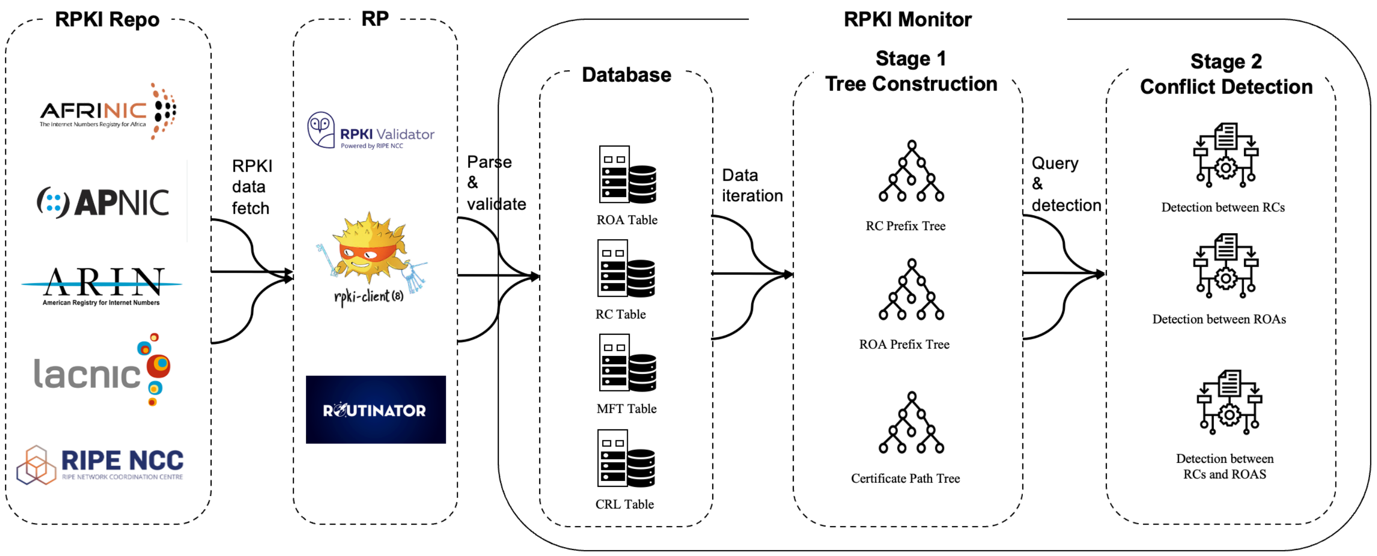
Type B: INR Overlapping



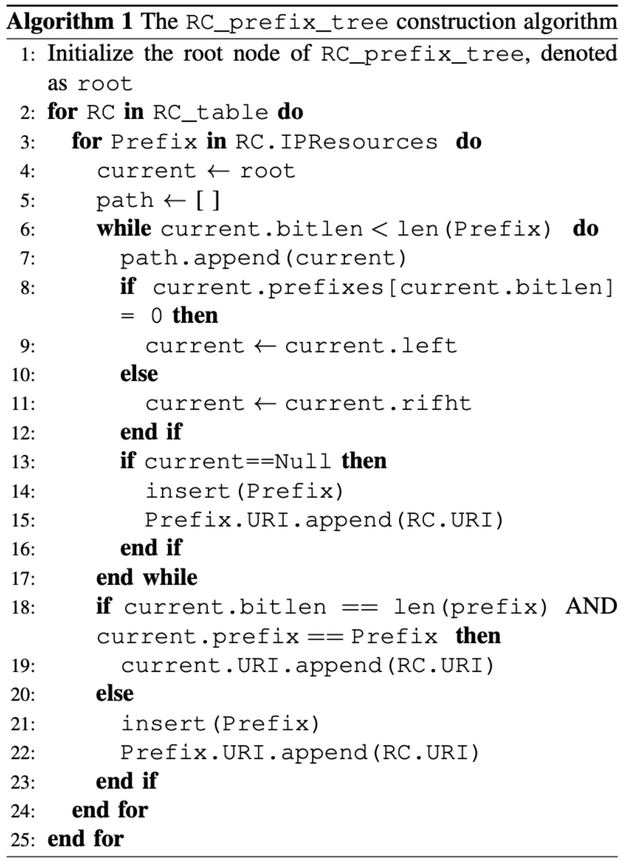
Outline

RPKI Monitor Framework Overview

- ➢ Fetch RPKI data objects from repositories
- ➢ Database tables to store RPKI data: RC, ROA, MFT, CRL
- ➢ Construct prefix trees and certificate tree
- ➢ detect conflicts between RCs and ROAs



Stage 1： Prefix Tree Construction

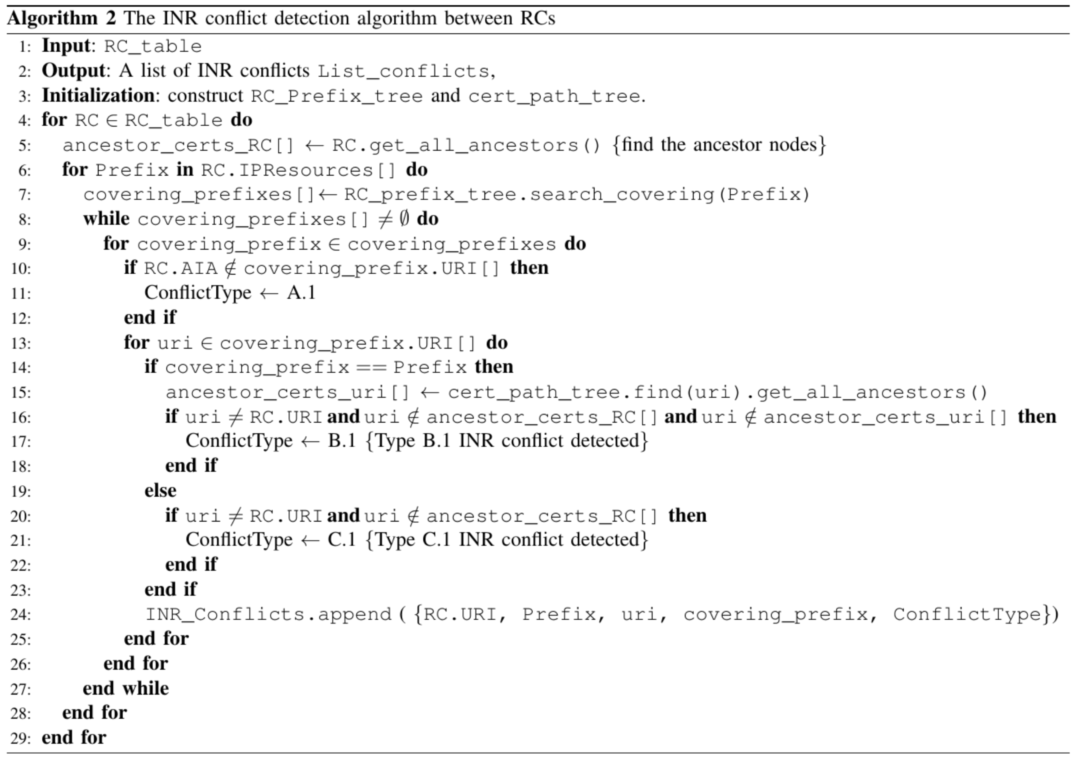


Algorithm 1 The RC_prefix_tree construction algorithm

```
1:  Initialize the root node of RC_prefix_tree, denoted
    as root
2:  for RC in RC_table do
3:    for Prefix in RC.IPResources do
4:      current ← root
5:      path ← [ ]
6:      while current.bitlen < len(Prefix) do
7:        path.append(current)
8:        if current.prefixes[current.bitlen]
          = 0 then
9:          current ← current.left
10:       else
11:         current ← current.rifht
12:       end if
13:       if current==Null then
14:         insert(Prefix)
15:         Prefix.URI.append(RC.URI)
16:       end if
17:     end while
18:     if current.bitlen == len(prefix) AND
        current.prefix == Prefix then
19:       current.URI.append(RC.URI)
20:     else
21:       insert(Prefix)
22:       Prefix.URI.append(RC.URI)
23:     end if
24:   end for
25: end for
```

- Construct the prefix trees for RCs and ROAs based on the radix tree structure
  - each node is identified by an IP prefix
  - each node is associated a list of URIs which identify RCs(ROAs) that encompass the prefix.
- Construct an auxiliary certificate path tree to facilitate the forming of certificate chain

Stage 2： Conflict Detection



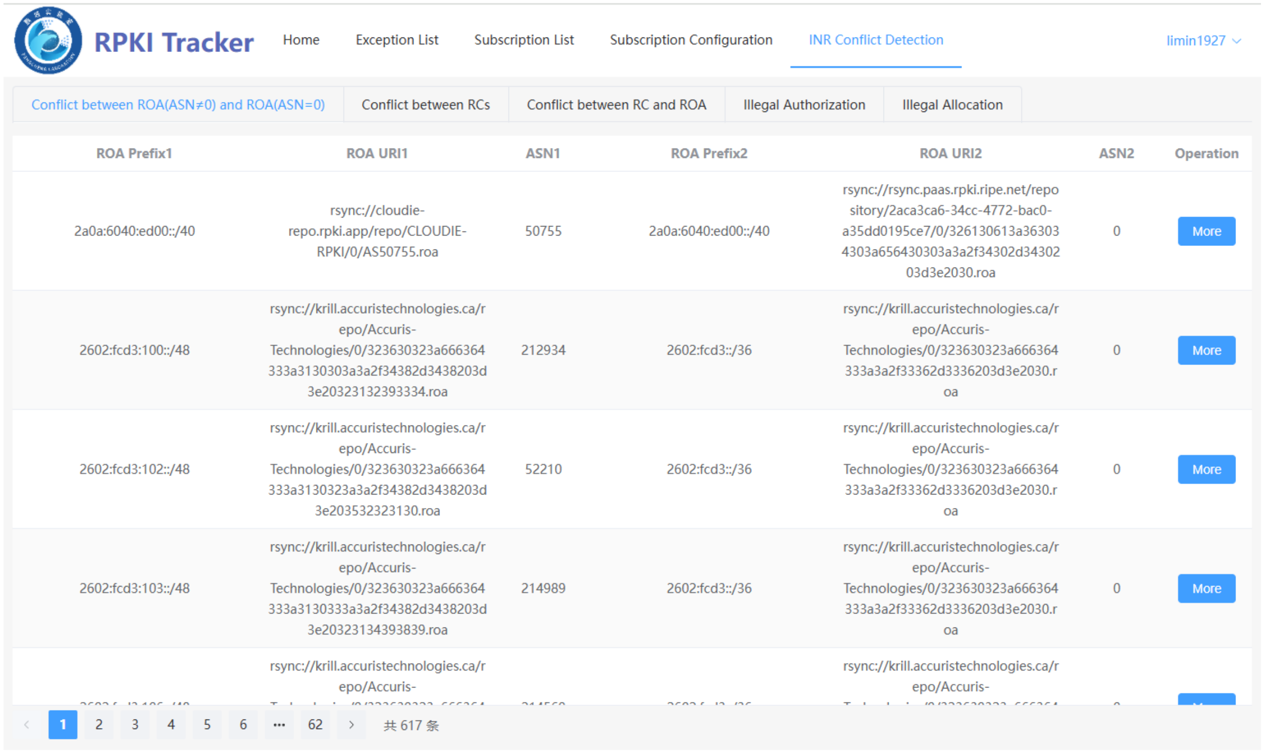
Algorithm 2 The INR conflict detection algorithm between RCs

```
Input: RC_table
Output: A list of INR conflicts List_conflicts,
Initialization: construct RC_Prefix_tree and cert_path_tree.
for RC $\in$ RC_table do
  ancestor_certs_RC[] $\leftarrow$ RC.get_all_ancestors() {find the ancestor nodes}
  for Prefix in RC.IPResources[] do
    covering_prefixes[] $\leftarrow$ RC_prefix_tree.search_covering(Prefix)
    while covering_prefixes[] $\neq \emptyset$ do
      for covering_prefix $\in$ covering_prefixes do
        if RC.AIA $\notin$ covering_prefix.URI[] then
          ConflictType $\leftarrow$ A.1
        end if
        for uri $\in$ covering_prefix.URI[] do
          if covering_prefix == Prefix then
            ancestor_certs_uri[] $\leftarrow$ cert_path_tree.find(uri).get_all_ancestors()
            if uri $\neq$ RC.URI and uri $\notin$ ancestor_certs_RC[] and uri $\notin$ ancestor_certs_uri[] then
              ConflictType $\leftarrow$ B.1 {Type B.1 INR conflict detected}
            end if
          else
            if uri $\neq$ RC.URI and uri $\notin$ ancestor_certs_RC[] then
              ConflictType $\leftarrow$ C.1 {Type C.1 INR conflict detected}
            end if
          end if
          INR_Conflicts.append({RC.URI, Prefix, uri, covering_prefix, ConflictType})
        end for
      end for
    end while
  end for
end for
```

- Detect the conflicts between different objects
  - RC-RC
  - RC-ROA
  - ROA-ROA

RPKI Monitor: system implementation

We implement the RPKI Monitor and integrate it into the RPKI Tracker Platform



ROA Prefix1	ROA URI1	ASN1	ROA Prefix2	ROA URI2	ASN2	Operation
2a0a:6040:ed00::/40	rsync://cloudie-repo.rpki.app/repo/CLOUDIE-RPKI/0/AS50755.roa	50755	2a0a:6040:ed00::/40	rsync://rsync.paas.rpki.ripe.net/repository/2aca3ca6-34cc-4772-bac0-a35dd0195ce7/0/326130613a363034303a656430303a3a2f34302d3430203d3e2030.roa	0	More
2602:fcd3:100::/48	rsync://krill.accuristechnologies.ca/repo/Accuris-Technologies/0/323630323a666364333a3130303a3a2f34382d3438203d3e20323132393334.roa	212934	2602:fcd3::/36	rsync://krill.accuristechnologies.ca/repo/Accuris-Technologies/0/323630323a666364333a3a2f33362d3336203d3e2030.roa	0	More
2602:fcd3:102::/48	rsync://krill.accuristechnologies.ca/repo/Accuris-Technologies/0/323630323a666364333a3130323a3a2f34382d3438203d3e203532323130.roa	52210	2602:fcd3::/36	rsync://krill.accuristechnologies.ca/repo/Accuris-Technologies/0/323630323a666364333a3a2f33362d3336203d3e2030.roa	0	More
2602:fcd3:103::/48	rsync://krill.accuristechnologies.ca/repo/Accuris-Technologies/0/323630323a666364333a3130333a3a2f34382d3438203d3e20323134393839.roa	214989	2602:fcd3::/36	rsync://krill.accuristechnologies.ca/repo/Accuris-Technologies/0/323630323a666364333a3a2f33362d3336203d3e2030.roa	0	More
[illegible]	rsync://krill.accuristechnologies.ca/repo/Accuris-	[illegible]	[illegible]	rsync://krill.accuristechnologies.ca/repo/Accuris-	[illegible]	

‹ 1 2 3 4 5 6 ••• 62 › 共 617 条

- ✓ INR conflicts are detected and recorded in real-time
- ✓ categorized by conflict type
- ✓ conflict detail provided
  - ✓ prefix
  - ✓ AS
  - ✓ URI
  - ✓ conflict duration
- ✓ intelligence disclosure
  - ✓ user can subscribe conflict information
  - ✓ daily digest will then be delivered to users via email

RPKI Monitor: limitations

Conflicts are not necessarily errors, and it is hard to tell A from B!

conflicts may be caused by different reasons

business driven configurations

- anycast
- multi-homing
- temporary conflicts during transfer
- …

V.S.

errors or attacks

- mis-configuration
- intentional attacks
- uni-lateral misbehavior of CA
- …

error determination and recovery relies on some
out-of-band information that hard to obtain

- technical
  - ISP configurations
  - routing strategy
  - …
- commercial
  - customer-provider relationships
  - resource allocation strategy
  - resource transfer plan
  - …

RPKI Monitor: Can-do & Cannot-do (in current state)

Can-do

- continuous and comprehensive detection, monitoring and analysis on INR conflicts
- INR conflict intelligence disclosure platform
- a reference value for RPKI troubleshooting

Cannot-do

- accurate error determination
- 100% effective error recovery suggestion
- users may need to combine their private domain information to further determine whether our suggestions work

RPKI can function like a CT scanner, which can provide some medical image information to the doctor

But it is ultimately up to the doctor to make the diagnosis and treatment plan

RPKI Monitor: cooperation with industries

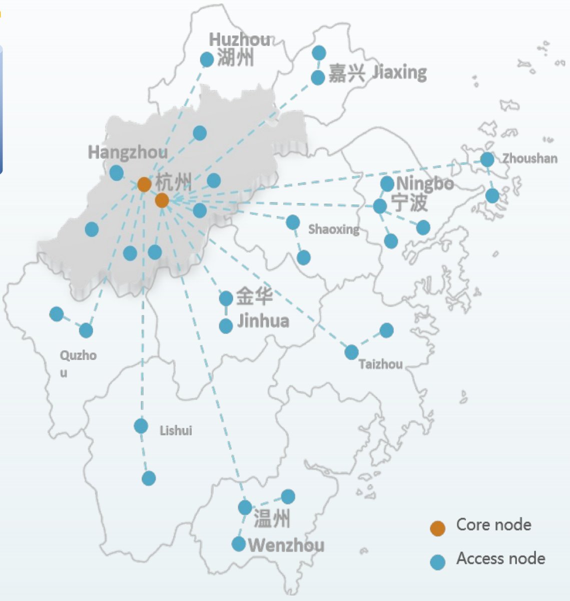
NNIX is the first National IXP approved by the MIIT of China. The RPKI Monitor platform has been deployed and applied in NNIX to provide continuous and real-time INR conflicts and RPKI/BGP anomalies monitoring services

NNIX Basic Information

Network Scale

288	49.8T	8.7T
Peering	Band width	Peak Traffic

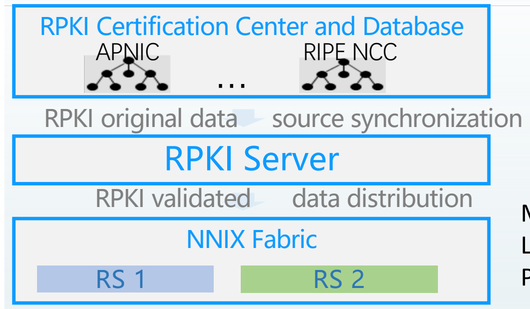
Network Node

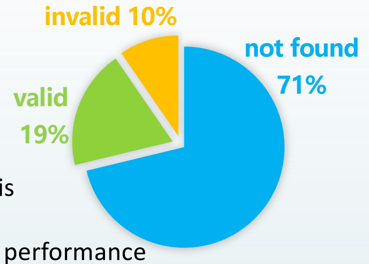


Services

- Peer-to-peer interconnection
- Multi-cloud interconnection
- Computing power scheduling
- Multidimensional products

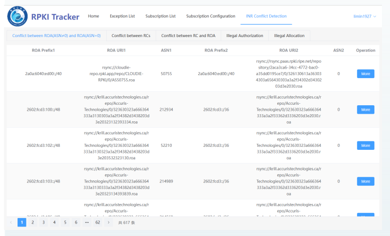
RPKI in NNIX





Manual analysis
Low efficiency
Poor real-time performance

RPKI Monitor Depolyed in NNIX



ROA Prefix1	ROA URI1	ASN1	ROA Prefix2	ROA URI2	ASN2	Operation
2a0a:6040:ed00::/40	rsync://cloudie-repo.rpki.app/repo/CLOUDIE-RPKI/0/AS50755.roa	50755	2a0a:6040:ed00::/40	rsync://rsync.paas.rpki.ripe.net/repository/2aca3ca6-34cc-4772-bac0-a35dd0195ce7/0/3261306133a363034303a656430303a3a2f34302d3430203d3e2030.roa	0	More
2602:fcd3:100::/48	rsync://krill.accuristechnologies.ca/repo/Accuris-Technologies/0/323630323a666364333a3130303a3a2f34382d3438203d3e20323132393334.roa	212934	2602:fcd3::/36	rsync://krill.accuristechnologies.ca/repo/Accuris-Technologies/0/323630323a666364333a3a2f33362d3336203d3e2030.roa	0	More
2602:fcd3:102::/48	rsync://krill.accuristechnologies.ca/repo/Accuris-Technologies/0/323630323a666364333a3130323a3a2f34382d3438203d3e203532323130.roa	52210	2602:fcd3::/36	rsync://krill.accuristechnologies.ca/repo/Accuris-Technologies/0/323630323a666364333a3a2f33362d3336203d3e2030.roa	0	More
2602:fcd3:103::/48	rsync://krill.accuristechnologies.ca/repo/Accuris-Technologies/0/323630323a666364333a3130333a3a2f34382d3438203d3e20323134393839.roa	214989	2602:fcd3::/36	rsync://krill.accuristechnologies.ca/repo/Accuris-Technologies/0/323630323a666364333a3a2f33362d3336203d3e2030.roa	0	More

‹ 1 2 3 4 5 6 ··· 62 › 共 617 条

- user feedbacks with expert knowledge
- hundreds of FPs reported

- 7*24 monitoring services
- 10000+ conflicts/anomalies reported

NATIONAL NOVEL INTERNET EXCHANGE

Outline

Overview of results

- **Type C** (INR overlapping) dominate the total incidents, accounting for **83.51%**
- **Type A** conflicts (illegal INR delegation) shows zero occurrences, suggesting well-controlling of this kind
- Conflicts involving **ASN = 0** rarely occur: **0.46%** for repetition and and **0.78%** for overlapping

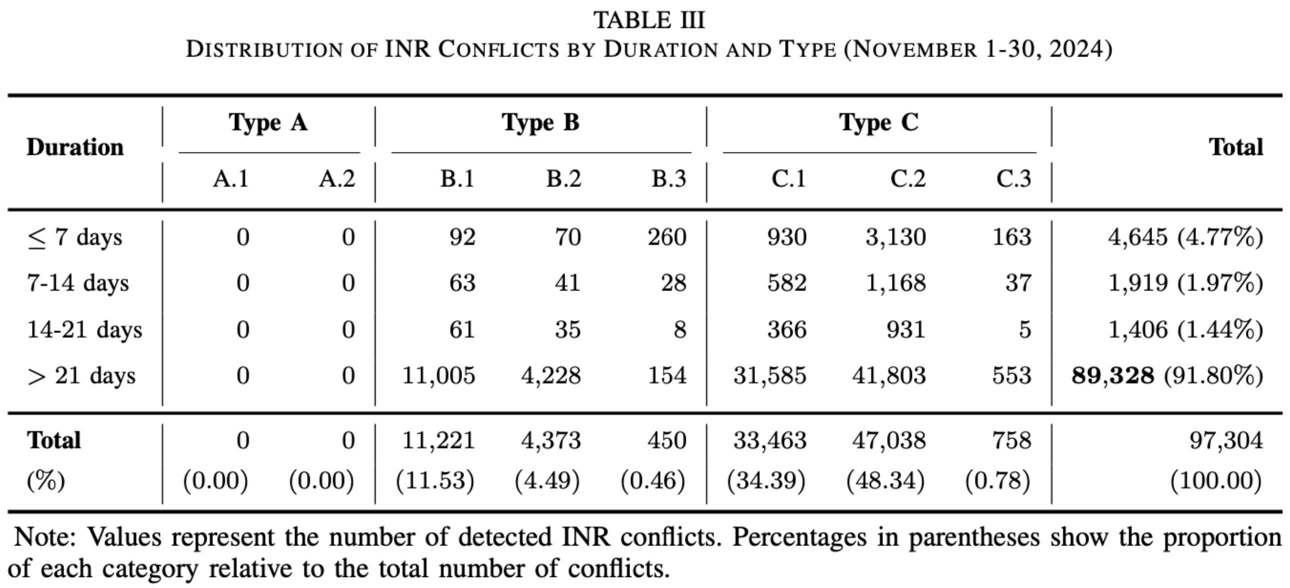
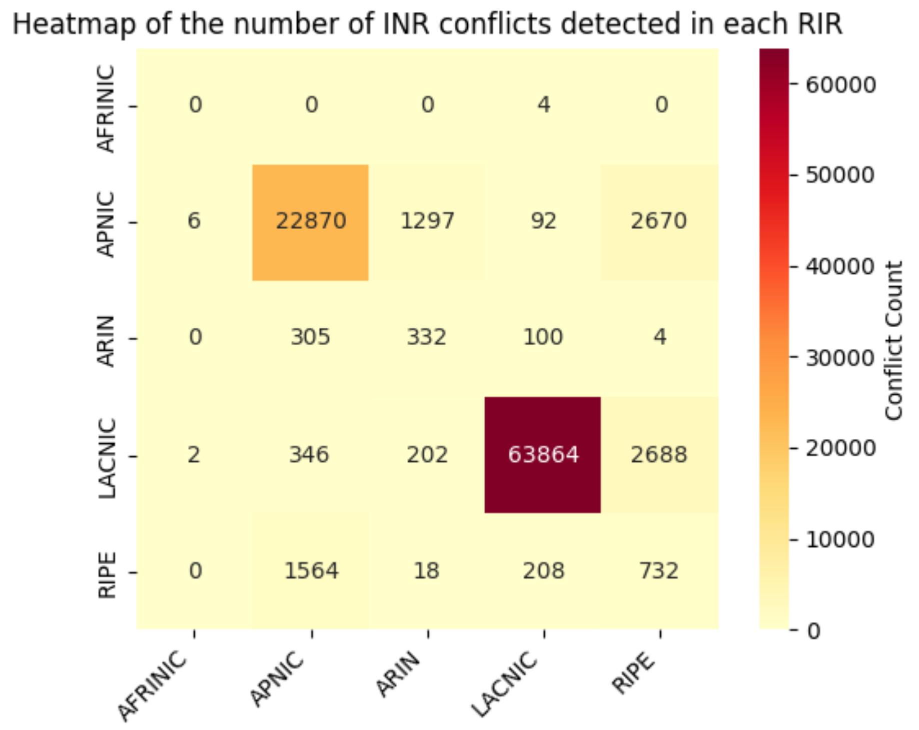


TABLE III
DISTRIBUTION OF INR CONFLICTS BY DURATION AND TYPE (NOVEMBER 1-30, 2024)

Duration	**Type A**		**Type B**			**Type C**			**Total**
	A.1	A.2	B.1	B.2	B.3	C.1	C.2	C.3	
$\leq$ 7 days	0	0	92	70	260	930	3,130	163	4,645 (4.77%)
7-14 days	0	0	63	41	28	582	1,168	37	1,919 (1.97%)
14-21 days	0	0	61	35	8	366	931	5	1,406 (1.44%)
$>$ 21 days	0	0	11,005	4,228	154	31,585	41,803	553	**89,328** (91.80%)
Total	0	0	11,221	4,373	450	33,463	47,038	758	97,304
(%)	(0.00)	(0.00)	(11.53)	(4.49)	(0.46)	(34.39)	(48.34)	(0.78)	(100.00)

Note: Values represent the number of detected INR conflicts. Percentages in parentheses show the proportion of each category relative to the total number of conflicts.

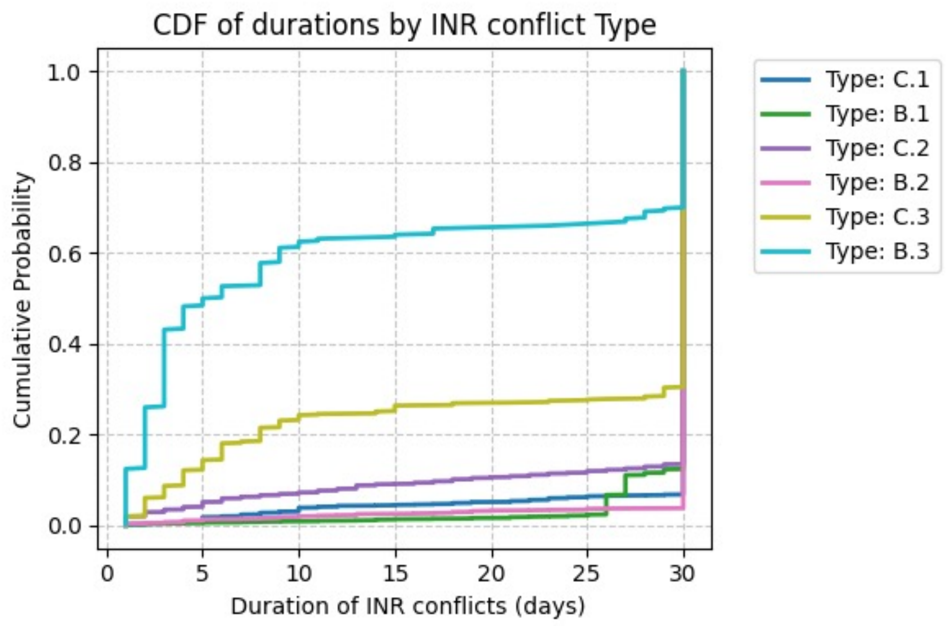
Regional Analysis


Heatmap of the number of INR conflicts detected in each RIR

	AFRINIC	APNIC	ARIN	LACNIC	RIPE
AFRINIC	0	0	0	4	0
APNIC	6	22870	1297	92	2670
ARIN	0	305	332	100	4
LACNIC	2	346	202	63864	2688
RIPE	0	1564	18	208	732

- **90.23%** intra-RIR conflicts v.s. **9.78%** inter-RIR conflicts
- **LACNIC**: the most severe region
  - 72.74% of the intra-RIR conflicts
  - 38.27 % of the inter-RIR conflicts
- **AFRINIC** contributes extremely low proportion of INR conflicts: **0.01%**

Temporal Analysis



- **Long-term existence:** over 90% last for above 3 weeks
- **Conflicts involving RCs and non-zero ASes** especially last for a long time
  - over 80% are not fixed during the whole month
- **Conflicts involving zero AS** show relatively shorter existence

Outline

Future Plan

- systemic issues beyond simple mis-configurations are ubiquitous in productive RPKI system, which are urgently need to be addressed in the future.
- rate the likelihood of a conflict truly being an error and provide suggested countermeasures
- **open-source the RPKI INR conflict tracking platform**, develop **unified and standardized APIs** for cross-RIR communication and cooperation. In line with that, **automated configuration tools** to fix conflicts can be also considered to be the next step.

Thank you for your listening!

Feedbacks from industries and Internet community are important to us!

liux15@pcl.ac.cn