RPKI Inter-Cache Synchronization
--Distributing RPKI Validated Cache in JSON over HTTPS

Di Ma
ZDNS and RPSTIR
madi@rpstir.net

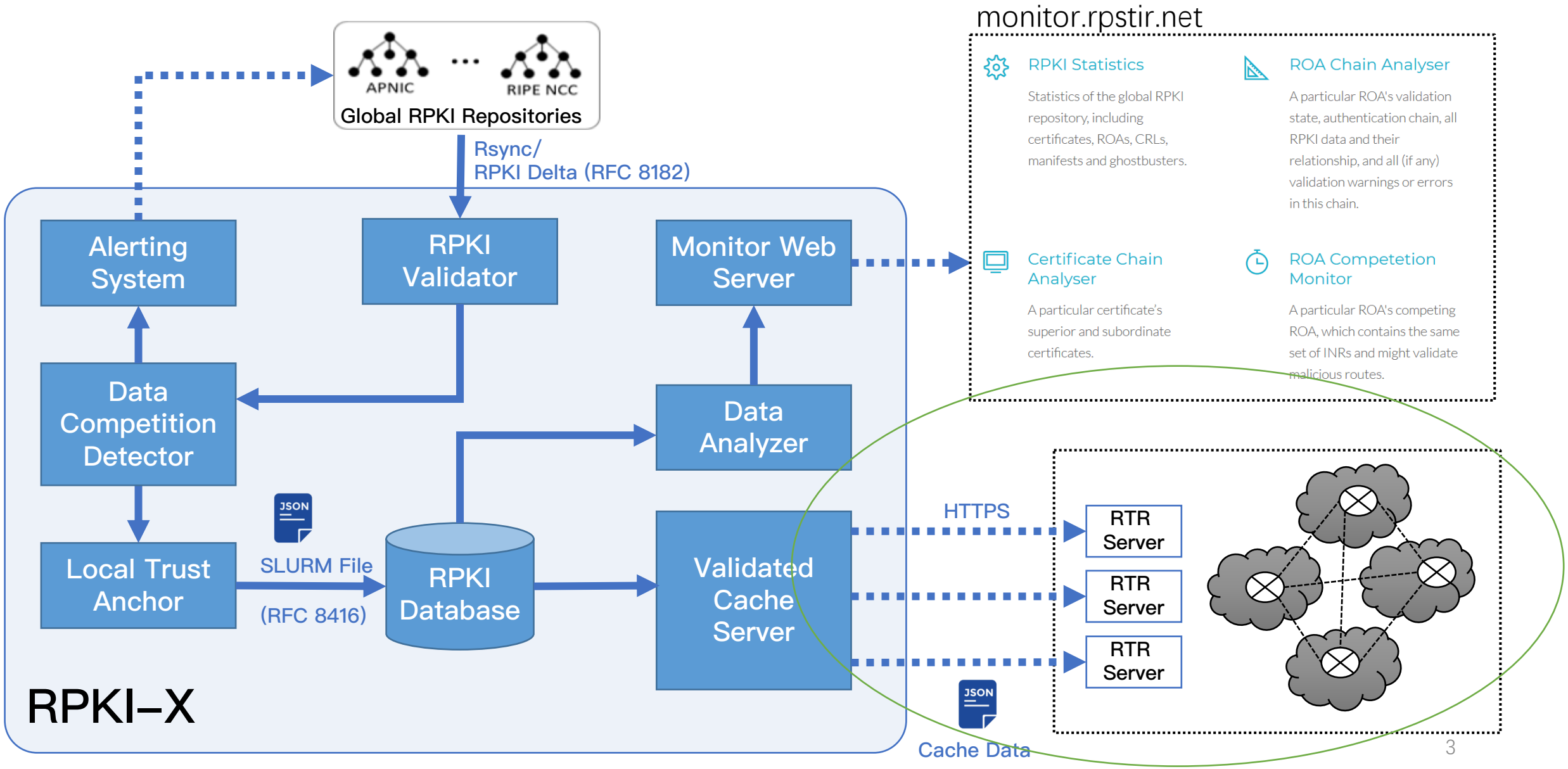
IETF 106, Singapore

RPKI Cache Deployment Scenarios, in RFC 8210

- Not all networks have to set up its own Relying Party system to do RPKI data synchronization and verification.

- RPKI validated cache is not necessarily transported to routers, but between two intermediate nodes arranged in a hierarchical scheme.

RPKI Cache Provisioning System, in Service



Considerations

- Data Format
  - Facilitate the data parse for both visualization and applications other than ROV
  - Take advantage of the SLURM (RFC 8416) to do local control

- Transport
  - Widely supported
  - Security enhancement to offer integrity protection
  - Protocol-Data-Unit independent

RPKI VC in JSON

- Validated Cache described in JSON format
- Piggybacked between two RPKI Cache Servers over HTTPS
- Incremental Update
  - SLURM-Filters (RFC 8416-Section 3.3) to delete
  - SLURM-Assertions (RFC 8416-Section 3.4) to insert

```
{
  "slurmVersion": 1,
  "validationOutputFilters": {
    "prefixFilters": [],
    "bgpsecFilters": []
  },
  "locallyAddedAssertions": {
    "prefixAssertions": [],
    "bgpsecAssertions": []
  }
}
```

SLURM File in JSON

RTR is over there, why bother to do this

- RTR is designed exclusively for provisioning RPKI cache for routers
  - PDU is bound to transport, which is difficult to add extensions to support VC synchronization management
  - Binary data unit can not be parsed by applications, which by the way do not support RTR generally
- HTTPs is widely deployed and data format independent
- RPKI VC Servers just need ONE API to do both synchronization and local control since JSON is employed by SLURM （RFC 8416）to override global RPKI data

RPKI Cache over HTTPS: PUSH

```
:method = POST
: scheme = https
: authority = vc.example.net
: path = /rpki-push
accept = application/rpki-message
content-length = 992
<992 bytes represented by the following json string>
{  "head": {   "operate": "update",
               "time": "2019-07-31 13:56:01",
               "fromversion": 1,
               "toversion": 2,
               "sha256withRSA": "<Base 64 of sha256withRSA>"  },
    "body": {   "slurmVersion": 1,
                "validationOutputFilters": {
                     "prefixFilters": [   {
                          "prefix": "198.51.100.0/24",
                          "asn": 64497,
                          "comment": "prefix and asn"  }   ]   },
                "locallyAddedAssertions": {
                    "prefixAssertions": [  {
                          "asn": 64496,
                          "prefix": "198.51.100.0/24",
                          "comment": "other route"    }    ]   }
  }   }
```

```
: status = 200
content-type = application/rpki-message
content-length = 28
 <28 bytes represented by the following json string>
{ "result":"ok",   "msg":""}


: status = 400
content-type = application/rpki-message
content-length = 59
 <59 bytes represented by the following  json string>
{ "result":"fail",   "msg":"sha256withRSA check failure"}
```

RPKI Cache over HTTPS: PULL

PULL for whole copy

```
:method = GET
:scheme = https
:authority = rp.example.net
:path = /rpki-pull
accept = application/rpki-message
```

PULL for incremental update

```
:method = GET
:scheme = https
:authority = rp.example.net
:path = /rpki-pull/version
accept = application/rpki-message
```

Standardization Considerations

- It is necessary to standardized RPKI VC synchronization.
  - RPKI VC synchronization could take place between different organizations and VC receiver may change its VC provider.

- A draft is coming soon and I hope it will be adopted as a WG item.
  - We keep an open mind to other proposed details ☺
  - I also note that Cloudflare has made the similar efforts.

Questions?