BBN Relying Party Software for the RPKI

Dr. Stephen T. Kent

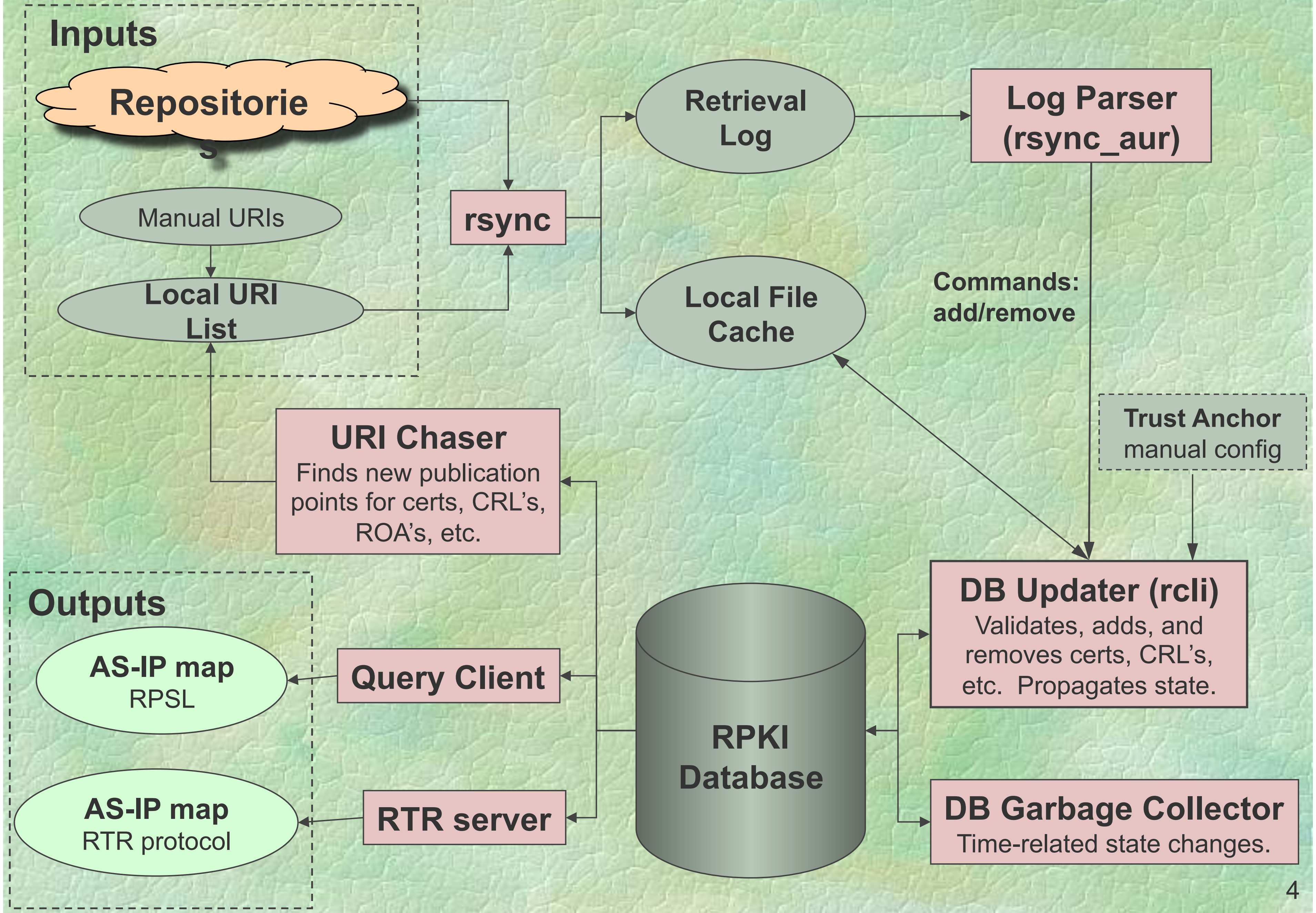
BBN TECHNOLOGIES

Relying Party Software Key Features (1/2)

- Local database of digitally certificates, CRLs, and RPKI signed objects
  - Fast cache of verified, immutable object values
  - Each file in the local repository (cache) is read and parsed only once
  - Straightforward, robust method for maintaining persistent state
- Efficient mechanism for handling certificates, CRLs, ROAs, manifests, and trust anchor objects
  - Almost complete implementation of local trust anchor support

Relying Party Software Key Features (2/2)

- Standards compliant (SIDR and PKIX), open source, multi-platform (some versions of Linux, FreeBSD, and OpenBSD, more later, incl. OS X)
- Automated background processing of object revocation and expiration
- Incremental, deferred validation of signed objects, tailored so that these objects can arrive in any order
    - Database maintains state so that expensive operations like signature verification and hash computation are performed only once


Inputs
Repositories
Manual URIs
Local URI List
rsync
Retrieval Log
Log Parser (rsync_aur)
Local File Cache
Commands: add/remove
Trust Anchor manual config
URI Chaser
Finds new publication points for certs, CRL's, ROA's, etc.
DB Updater (rcli)
Validates, adds, and removes certs, CRL's, etc. Propagates state.
Outputs
AS-IP map
RPSL
Query Client
AS-IP map
RTR protocol
RTR server
RPKI Database
DB Garbage Collector
Time-related state changes.

Remote Synchronization

- Synchronization is built on rsync
  - Static configuration file of remote repositories
  - Dynamic configuration file of CRLDPs generated by the "chaser" component
- All actions logged
- Local repository (file system) is updated based on rsync actions
- Software also maintains certain protected directories that rsync does not touch
  - Extracted versions of embedded EE certificates
  - Backup copy of manifest for each directory
  - Trust anchors (global and local)
- Run as a cron job; can also be run manually

Database Intake

- Triggered by rsync completion
- Parses rsync log file and performs indicated actions
- Performs syntactic validation of all objects before they are entered into the database
- Embedded EE certificates in ROAs and manifests are extracted and handled independently
  - Linked to the original object in the DB
- Trust anchors are added out of band

Database Structure

- One table for each type of object
- Adding objects triggers object-specific actions
  - Adding a certificate triggers path discovery
  - Adding a CRL revokes all the certificates in the DB named by that CRL
- Software “glue” layer abstracts table layout so that changes in DB structure (e.g., recent addition of support for CTA) are isolated to a small set of interfaces written in C
- DB locking is used to prevent collisions between asynchronous software components
- DB presents an external interface that implements the RTR protocol
- Highly tuned and optimized for optimal performance

Garbage Collector

- Handles time-related state changes
  - Certificate expiration
  - ROA (EE certificate) expiration
  - CRL staleness
  - Manifest staleness
- Run asynchronously as a cron job

URI Chaser

- Scans for CRLDPs (certificate extension)
- Compacts the list of CRLDPs to a minimal subset
  - Reduced to smallest set that fetches from all CRL publication points in order to minimize network traffic
- Updates rsync’s dynamic configuration file
- An asynchronous process

RPSL Output Generation

- Database query client extracts a raw set of "plausibly valid" ROAs
- Applies <u>user-configurable </u>filters
  - Stale CRL OK?
  - Stale Manifest OK?
  - Expired certificate in path OK?
  - Superseded manifest OK?
- All filters have default settings based on our understanding of most likely use case
- Generates RPSL based on filtered output
- Can be run synchronously or asynchronously
- Can also be used to perform generalized database queries without a user needing to learn SQL

Router Protocol Support

- Newly proposed protocol (draft-ietf-sidr-rpki-rtr-01) for communication between an RPKI server and a router (within an AS)
- The protocol assumes a server that
  - Fetches certificates, CRLs, and signed objects from the RPKI repository system
  - Processes these objects to maintain a local cache
  - Sends messages to routers to notify of cache updates, and replies to queries with <prefix, ASN list
- The RP software provides a suitable server cache

Current System Performance

- Tested the system using simulated repository data generated from RIR "profiles"
  - 9,932 CA certificates
    13,292 EE (embedded) certificates
    6,646 CRLs
    6,646 ROAs
    6,646 manifests
    43,162 objects in all => 47 minutes 26 seconds
- Whole Internet deployment (~300,000 objects) => ~ 5.6 hours (one time cost of initializing DB)
- Typical daily update (3,000 objects) => less than five minutes
- We're working to improve these numbers

Planned Software Enhancements

- Complete local trust anchor management, as defined in draft-reynolds-rpki-ltamgmt-00.tx
- Complete port to Mac OS X, FreeBSD (7), OpenBSD
- Track SIDR decision on trust anchor configuration, and support accordingly
- Support new RPKI signed objects as they are defined, e.g., the “Ghost Buster” record
- Enable parallel rsync fetches, back off and retry
- Improve performance, fix bugs, …

Questions?