NDN ROUTING SECURITY

Lan Wang, Beichuan Zhang

Routing Security

- Required: authenticity and integrity of routing information
  - Link state routing: LSAs are originated by the routing process authorized to do so and have not been modified.
  - Hyperbolic routing: hyperbolic coordinates are coordinates for the associated nodes and prefixes
- Not required: confidentiality of routing information
- Solution: routing data is signed by originating router and verified by receivers based on trust model.

Example: NLSR Trust Model [1]

- NLSR's trust model follows network management structure in a single network.
  - The entire network has a root key, the trust anchor (pre-configured at every router).

```
                                    root
                                     |
                      +--------------+--------------+
                     site1                        site2
                      |                             |
            +---------+---------+                   +
        operator1            operator2          operator3
            |                    |                  |
      +-----+-----+        +-----+-----+      +-----+-----+--------+
   router1     router2  router3     router4 router5     router6  router7
      |           |        |           |       |           |        |
      +           +        +           +       +           +        +
     NLSR        NSLR     NSLR        NSLR    NSLR        NSLR     NSLR
```

[1] AKM M. Hoque, S. O. Amin, A. Alyyan, B. Zhang, L. Zhang, and L. Wang. *NLSR: Named-data link state routing protocol*. In ACM SIGCOMM ICN Workshop, 2013.

BigCo/NetOps/SFpop/OSPF/rtr731/pid345*/LSP#678*

The name in one Link State Packet generated by the SFpop IGP routing process on rtr731.

signed by

BigCo/NetOps/SFpop/OSPF/rtr731/pid345

The name of the routing process cert (given to the process when the router creates it). This name must match the bold part of each packet's name.

signed by

BigCo/NetOps/SFpop/RTR731

The name of the router cert (given to the router when it's configured).

signed by

* BigCo/NetOps/SFpop/config/employee975

The name of the cert of the employee who last configured the router. (This key signs the entire router config as well as the router's signing key.)

signed by

* BigCo/NetOps/SFpop/config

The name of the cert that is the root of trust for SFpop router configuration.

Trust Schema

k4 = my.config.root	**BigCo/NetOps/SFpop/config/key**
k3 = k4 +"empl"+ n	**BigCo/NetOps/SFpop/config/empl/975/key**
k2 = k3[-4] +"rtr"+ n	**BigCo/NetOps/SFpop/rtr/72/key**
k1 = k2[-3] +"OSPF"+ k2[2-1] +"pid"+ n	**BigCo/NetOps/SFpop/OSPF/rtr/72/pid/345/key**
pkt = k1 +"LSP"+ n	**BigCo/NetOps/SFpop/OSPF/rtr/72/pid/345/LSP/678**

Usage

if (validTrustChain(pkt, schema) && signatureValid(pkt))
  process the packet

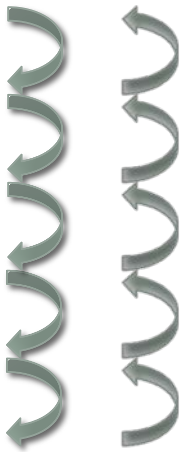
Since schema is just lexical constraints on key names, validation normally only has to check that key name is appropriate for data name.

Only have to validate chain & signature for a key once.

Signing and Verification

Entity	Name
Root key	/<network>/key
Site key	/<network>/<site>/key
Operator key	/<network>/<site>/<operator>/key
Router key	/<network>/<site>/<router>/key
NLSR key	/<network>/<site>/<router>/NLSR/key
Data	/<network>/NLSR/LSA/<site>/<router>/<type>/<ver>

sign verify



NLSR Security Configuration

```
security
{
  validator
  { ...
   rule
    {
      id "NSLR LSA Rule"
      for data
      filter
      {
        type name
        regex ^[^<NLSR><LSA>]*<NLSR><LSA>
      }
      checker
      {
        type customized
        sig-type rsa-sha256
        key-locator
        {
          type name
          hyper-relation
          {
            k-regex ^([^<KEY><NLSR>]*)<NLSR><KEY><ksk-.*><ID-CERT>$
            k-expand \\1
            h-relation equal
            p-regex ^([^<NLSR><LSA>]*)<NLSR><LSA>(<>*)<><><>$
            p-expand \\1\\2
          }
        }
      }
}
```

```
…
rule
    {
      id "NSLR Hierarchical Rule"
      for data
      filter
      {
        type name
        regex ^[^<KEY>]*<KEY><ksk-.*><ID-CERT><>$
      }
      checker
      {
        type hierarchical
        sig-type rsa-sha256
      }
    }

    trust-anchor
    {
      type file
      file-name "root.cert"
    }
  }
 ; cert-to-publish "site.cert"  ; optional, containing the site certificate.
 ; cert-to-publish "operator.cert" ; optional, containing the operator cert.
 cert-to-publish "router.cert"  ; a file containing the router certificate.
}
```

Issues in NLSR Security Implementation (1)

- Key generation and signing.
  - Whenever NLSR starts, it creates a new NLSR key.
  - NLSR signs the key using the router key.
    - what entity should have the authority to use the router key? A special launch process?
- Verification
  - Problem: timestamp of a received certificate may be later than the router's time (due to clock difference), which causes the router to drop the key and certificate
  - Current solution: when signing a key, the timestamp on the certificate is earlier than the actual clock time
  - Is this the right solution?

Issues in NLSR Security Implementation (2)

- NLSR key rollover
  - When NLSR restarts, it generates a new key. How do other routers know that from now on this key should be used rather than the previous key?
- Key revocation: an NLSR key (or router key etc.) is compromised and a new key needs to be used
  - Previous NLSR version used ChronoSync to distribute key names, which could solve this problem (and the previous one), but was taken out when new Validator was put in.

Issues in NLSR Security Implementation (3)

- Key retrieval and key name
  - key is retrieved after NLSR Data packet is received (if the key has not been retrieved)
  - Currently Interests for keys are broadcast (no FIB entries for the keys until routing table is built)
  - Below are alternatives:
    - use ChronoSync to distribute key names: the keys still need to have a broadcast prefix since ChronoSync doesn't actually send the keys in its data packets (unless ChronoSync always piggybacks the keys in its data packets).
    - append key to data packet when a node replies with NLSR data: requires composite packet format, but makes the packet bigger than necessary if the receiver already has the key
    - sends Interest for key to the neighbor that previously replied with the NLSR data: requires NLSR to know which face the data came in and send key Interest to that face