NDN Security Library

Yingdi Yu, Alexander Afanasyev,
Lixia Zhang
UCLA
August 2013
Structure

• Keychain
  – the interface to application
  – consists of three modules
    • Identity Manager
    • Policy Manager
    • Encryption Manager

• Identity Manager
  – manages identities, asymmetric keys, certificates
  – packet signing

• Policy Manager
  – manages verification/signing policies

• Encryption Manager
  – manages symmetric keys
  – packet encryption/decryption

• Packet verification is done by Keychain
• Leverage system keychain services (e.g., Mac OS keychain) for secret storage.
Figure

NDN Security Library

Identity Manager

Public Key Certificate Store

Private Key Store

Identity

Security Transform

Verify

Encrypt

Security Transform

Sign

Decrypt

Policy Manager

Verification Policy

Signing Policy
Private Storage

• Leverage system key chain service for secret storage
  – Mac OS X: keychain
  – Linux: GNOME Keyring

• Functionality
  – generate/store asymmetric key pairs
  – sign packet in black box
  – generate/store symmetric key for local encryption
    • encrypt/decrypt policy files
    • encrypt/decrypt symmetric keys which are used for packet encryption/decryption
Identity

- Identity represents the subject of a certificate
- Who may have an identity?
  - Organization
    - /ndn/ucla.edu/
    - /ndn/ucla.edu/cs/
  - User: corresponds to a user account on a system
    - /ndn/ucla.edu/cs/yingdi/
    - /ndn/ucla.edu/router1
  - Application: corresponds to an application run by a particular user
    - /ndn/ucla.edu/cs/yingdi/chronoshare/
    - /ndn/ucla.edu/router1/nlsr/
  - Application may define identities under its own namespace if necessary
- Sign-by-Identity
  - When signing a packet, one specify which identity should be used, and Identity Manager will automatically determine the signing key and the key locator (name of the certificate).
Identity & Keys

- An identity may have multiple pairs of keys
  - Two types of keys
    - Key Signing Key (KSK)
      - sign keys belonging to the same identity
    - Data Signing Key (DSK)
      - sign data generated by the identity
      - sign keys belonging to other identities
- Each identity has a default key pair
- An identity may have multiple DSKs
  - User may have multiple devices sharing the same user identity
- Some identities (e.g., application) may have ZSK only
- Some identities (e.g., organization) may have multiple KSKs for redundancy
Example of identities & keys

/ndn/DSK-1
/ndn/ucla.edu/KSK-1
/ndn/ucla.edu/DSK-11
/ndn/ucla.edu/KSK-1
/ndn/ucla.edu/DSK-11
/ndn/ucla.edu/KSK-2
/ndn/ucla.edu/KSK-2
/ndn/ucla.edu/DSK-1
/ndn/ucla.edu/KSK-1
/ndn/ucla.edu/DSK-11
/ndn/ucla.edu/KSK-3
/ndn/ucla.edu/DSK-13
/ndn/ucla.edu/KSK-5
/ndn/ucla.edu/DSK-14
/ndn/ucla.edu/chronshare/KSK-3
/ndn/ucla.edu/chronshare/KSK-5
/ndn/ucla.edu/chronshare/DSK-13
/ndn/ucla.edu/chronshare/DSK-14

macbook
chronoshare
mac
chronoshare
Key & Certificate

- A public key may have multiple certificates
  - signed by different parties
  - signed by different keys of the same party
- Each public key has a default certificate
- A NDN Certificate is a Data packet

![NDN Certificate Format]

```plaintext
Name
  <key_name>/ID-CERT/<id#>
Content
  DER encoded
  idCert ::= SEQUENCE {
    validity
    subject
    subjectPubKeyInfo
    extension}
  validity ::= SEQUENCE {
    notBefore
    notAfter }
  subject ::= SEQUENCE {
    subjectDescription }
  subjectPubKeyInfo ::= SEQUENCE {
    algorithm
    keybits }
  extension ::= SEQUENCE {
    extensionItems }
KeyLocator
  issuer's certificate name
Signature
```
Identity Manager

• Consists of two module
  – Private Storage
    • private keys
    • in system keychain service
  – Public Storage
    • relations between
      – identities and keys
      – keys and certificates
    • in ~/.ndn-identity/identity.db

• One can sign-by-identity/key/certificate
  – determine signing keys and key locator (i.e., certificate name)

• A default identity for the user
  – configured in ~/.ndn-identity/default-identity
Policy

• Identity policy is supported for now
• Policy is based on NDN regular expression
• A policy consists of
  – verification rules
    • which packet must be verified, and who should be the legitimate signer
    • see definition later
  – verification exemption
    • which packet does not need to be verified
    • by regex, data with name matching against the regex is exempted from verification
  – signing rules
    • whether the signer is eligible to sign the packet
    • see definition later
  – signing inference
    • which identity should be used to sign the packet
    • by regex and corresponding expand, the result is the name of signing identity
  – trust anchors
    • a list of certificates treated as valid without validation
• Policies are stored in encrypted files
NDN regex

• Two levels of pattern matching
  – name level (NPM)
    • match a name against component pattern
  – component level (CPM)
    • match a component against string pattern

• CPM leverage standard regular expression
• NPM has dependency on CPM
  – match names with a component with a particular pattern
    • ../../ID-CERT/.../.../...
Name Level Pattern Matching

• Component
  – `<component_pattern>`
  – `component_pattern` is standard regular expression
  – `<>` is a alias of `<.*>` which match any component

• Anchors
  – `^<name_pattern>`
    • name whose first component matches `name_pattern`
  – `<name_pattern>$`
    • name whose last component matches `name_pattern`

• Modifiers
  – `<name_pattern>`*
    • repeat the `name_pattern` for 0 to more times
  – `<name_pattern>`+
    • repeat the `name_pattern` for 1 to more times
  – `<name_pattern>`?
    • repeat the `name_pattern` for 0 to 1 time
  – `<name_pattern>{n, m}`
    • repeat the `name_pattern` for `n` to `m` times
  – `[^<component_pattern1><component_pattern2>…]`
  – `[^<component_pattern1><component_pattern2>…]`
    • component_pattern set

• Back References
  – `(name_pattern)\n    • recall previously matched name pattern`
BNF Parsing Rule

\[
\begin{align*}
\text{<expression>} & : = \text{<pattern>} \mid ^{\text{<pattern>}}$ \mid ^{\text{<pattern>}}$
\text{<pattern>} & : = \text{<repeat_pattern>} \mid \text{<repeat_pattern>}\text{<pattern>}
\text{<repeat_pattern>} & : = \text{<sub_pattern>} \mid \text{<sub_pattern>}+ \mid \text{<sub_pattern>}* \mid \text{<sub_pattern>} \{\text{<repeat>}\}
\text{<repeat>} & : = \text{<num>} \mid \text{<num>} , \mid ,\text{<num>} \mid \text{<num>} ,\text{<num>}
\text{<sub_pattern>} & : = \text{<back_reference>} \mid \text{<component_set>}
\text{<back_reference>} & : = (\text{<pattern>})
\text{<component_set>} & : = \text{<component>} \mid [\text{<component_list>}][^\text{<component_list>}]$
\text{<component_list>} & : = \text{<component>} \mid \text{<component>}\text{<component_list>}
\text{<component>} & : = \text{<<regex>>}
\text{<regex>} & : = \text{standard regular expression}
\text{<num>} & : = \text{numbers}
\end{align*}
\]
Definition Rules

- 5 components
  - `data_regex`: `^(<>*)<DNS>(<>)<>*$`
    - `/ndn/ucla.edu/cs/yingdi/DNS/chronoshare/…/`
  - `data_expand`: `\1\2`
    - `/ndn/ucla.edu/cs/yingdi/chronoshare/`
  - `signer_regex`: `^(<>*)<DNS>(<>*)<NDNCERT><>*$`
    - `/ndn/ucla.edu/DNS/cs/yingdi/NDNCERT/…/`
  - `signer_expand`: `\1\2`
    - `/ndn/ucla.edu/cs/yingdi/`
  - `relation`: `>=`
    - `/ndn/ucla.edu/cs/yingdi/chronoshare/ >= /ndn/ucla.edu/cs/yingdi/`

- When verifying/signing a packet, data name and signer name must comply with the corresponding verification rules and signing rules
- A policy may have multiple rules, the order of rules matters!
  - more specific rules should go first
- One may also use exclusion “[^...]” to explicitly make rules exclusive
Verification Steps

- is there a verification policy for the namespace?
  - Yes
  - does the data name and key locator comply with the policy?
    - Yes
    - is the key locator a trust anchor?
      - Yes
      - Verification fails
      - No
      - did we reach the limit of verification?
        - Yes
        - Start signature verification
        - No
        - Verification fails
    - No
  - No
    - is there a verification-exempt policy for the namespace?
      - Yes
      - Accept the packet
      - No
      - Verification fails

Fetch the certificate pointed by key locator

Fetch the certificate as an Data packet