Packet Validation in the Network Environments

Yingdi Yu
UCLA
Packet Authentication

• How to authenticate a data packet containing the electricity usage of a room at certain time?

• Data is signed, but how to verify the signature?
  – How to get the signer’s public key?
  – How to authenticate the signer?
  – Why the signer should be trusted?
  – Should the signer be trusted at this moment?
Data & Certificate

- Retrieved as data packets
  - public keys are just another type of content

- Data packets are similar to certificates
  - data is signed

- Data packets are incomplete certificates
  - no signature validity period
  - no signature revocation information

- Current solution:
  - put validity period & other extensions in content

- Ideal solution:
  - extend SignatureInfo

---

<table>
<thead>
<tr>
<th>Name: Certificate name</th>
</tr>
</thead>
<tbody>
<tr>
<td>MetaInfo:</td>
</tr>
<tr>
<td>Content: (DER encoded)</td>
</tr>
<tr>
<td>ValidityPeriod:</td>
</tr>
<tr>
<td>NotBefore</td>
</tr>
<tr>
<td>NotAfter</td>
</tr>
<tr>
<td>PublicKeyInfo:</td>
</tr>
<tr>
<td>Extensions:</td>
</tr>
<tr>
<td>SignatureInfo:</td>
</tr>
<tr>
<td>SignatureType:</td>
</tr>
<tr>
<td>KeyLocator:</td>
</tr>
<tr>
<td>SignatureValue:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name: Certificate name</th>
</tr>
</thead>
<tbody>
<tr>
<td>MetaInfo:</td>
</tr>
<tr>
<td>Content:</td>
</tr>
<tr>
<td>PublicKeyInfo: (Still X509 format)</td>
</tr>
<tr>
<td>SignatureInfo:</td>
</tr>
<tr>
<td>SignatureType:</td>
</tr>
<tr>
<td>KeyLocator:</td>
</tr>
<tr>
<td>CriticalExtension:</td>
</tr>
<tr>
<td>NonCriticalExtension:</td>
</tr>
<tr>
<td>SignatureValue:</td>
</tr>
</tbody>
</table>
Naming

- Every data is named, what is the name of certificate?

- A certificate binds a key to a namespace (identity)
  - e.g., /<namespace>/[KeyId]
    - absolute KeyId: globally unique, e.g., key hash
    - relative KeyId: uniquely identify a key under the namespace, e.g., SeqNo

- Application interprets the namespace as some real world identity
  - in BMS, “/bms/boelter/4805/electrical” is interpreted as a sensor in the Room 4805 of Boelter Hall at UCLA
  - in openHealth, “/ucla/haitao/ndnex/dvu” is interpreted as a health data publisher of a user “/ucla/haitao”

- Certificate name may include version number
  - different signature versions (Key rollover)
Public Key Fetching/Provisioning

• Express an interest using the cert name in KeyLocator
  – certificate name of signer’s public key (w/o version)

• Certificate is published somewhere
  – current solution:
    • published as NDN DNS record
      – /ndn/ucla/KEY/yingdi/ksk-123/ID-CERT/%01
    • published through repo
  – issue: prefix aggregation
    • demux interest for certificate introduces extra name components
      in cert name
      – /ndn/KEY/ucla/yingdi/ksk-123/ID-CERT/%01
      – /ndn/ucla/yingdi/KEY/ksk-123/ID-CERT/%01

• General certificate infrastructure? or app-specific certificate infrastructure?
Signer Authentication

• Construct a chain of trust

/bms/KEY/dsk-376

/bms/boelter/KEY/dsk-821

/bms/boelter/4805/KEY/dsk-433

/bms/boelter/4805/electrical/20150101

Trust anchor
The origin of trust

Intermediate Keys
The tool to verify signature

Policy
The rule to regulate the chain of trust
Validation Framework

- Origin Validation Request
- Policy Checker
  - Key Retrieval
  - Internal Validation Request
  - Check Signature
- Authenticator
- Success

Info Manager
- Unverified Info
  - Info Fetcher
  - Data Buffer
- Trusted Info
  - Trust Anchors
  - Certificate Cache

- Auth Info
- Failure
- Failure
- Failure
- Failure
Policy

- Conditions on the SignatureInfo

- SignatureType
  - some data may require certain type of signature
    - algorithm
    - key size

- KeyLocator
  - some data must be signed by certain parties

- ValidityPeriod
  - signature must be valid at certain timestamp
Policy Rules

• A rule consists of
  – a filter
  – a set of checkers

• Filter
  – which packet should be checked by the rule

• Checker
  – the conditions that the packet’s SigInfo must meet
  – could be more than one sets of valid conditions
  – pass one checker, pass the rule
  – fail all checkers, fail the policy checking

• Order of rules matters
  – packet will be checked by the first matched rule
  – rules with more specific filter should go first
Policy Language

- **Configurable**
  - allow apps/users to specify its own trust models

- **Interpretable**
  - library can build the validator according to configuration
  - entities with the same configuration file share the same trust model
    - if router can fetch the policy, router knows how to validate data

- **Easy to distribute**
  - can be published as data packet
  - data name can be fixed with implicit digest

```plaintext
rule {
    filter {
        packet-type data
        packet-name <bms><>*
    }
    checker {
        signature-type ecdsa-sha256
        min-key-size 256
        key-locator {
            k-pattern (<>*)<KEY>(<>*)<><ID-CERT> \1\2
            h-relation is-prefix-of
            p-pattern (<>*) \1
        }
    }
    checker {
        signature-type ecdsa-sha256
        min-key-size 256
        key-locator {
            k-pattern (<>*)<KEY>(<>*)<><ID-CERT> \1\2
            h-relation is-prefix-of
            p-pattern <bms>(<>*) \1
        }
    }
}
```
Multiple signature

- The same content object may be signed by different keys
  - certificates: the same <name, key> pair may be certified by different parties
    - in openHealth, a doctor's key may be signed by both patient & medical board of California in order to access the patient’s data
  - signature agility: different signing algorithms & key size

- Introduce a signature extension: OtherSignatureLocator
Validation Framework

Info Manager
- Unverified Info
  - Info Fetcher
- Data Buffer
- Trusted Info
  - Trust Anchors
  - Certificate Cache

Policy Checker

Authenticator

Origin Validation Request

Failure

Internal Validation Request

Success

Key Retrieval

Check Signature

Cache certificate

Failure

Auth Info

Failure

Internal Validation Request
Public key retrieval issues

• Slow start
  – retrieve keys one-by-one, multiple RTTs
  – may involve more data
    • multiple signatures

• Single point failure
  – validation fail if one key is missing
    • limited internet access
    • key provision failure

• Key Bundle: why not ask data provider to collect keys and publish them along with the data?
  – fate sharing
    • if data can be fetched, so do the keys
  – efficiency
    • if producer collect the keys once, it can benefit many verifiers
Key Bundle Requirements

• Publisher & consumer agree on the trust policy and trust anchor

• In BMS
  – single trust anchor
  – hierarchical policy

• While expressing interest for data, also expressing interests for proofs

For data
/bms/boelter/4805/electrical/20150201

For proof
/bms/boelter/4805/electrical/20150201/AUTH_INFO/hierarchy/3d4c89ef..

/bms/boelter/KEY/dsk-821

/bms/boelter/4805/KEY/dsk-433

/bms/boelter/4805/KEY/dsk-376
Validation Framework

Info Manager
- Unverified Info
  - Info Fetcher
- Data Buffer
- Trusted Info
  - Trust Anchors
  - Certificate Cache

Policy Checker

Authenticator

Success

Failure

Origin Validation Request

Authenticated Info

Validation Request

Internal Validation Request

Check Signature

Cache certificate

Key Retrieval

Unverified Info

Failures
Signature Verification

- Start when reaching an pre-authenticated key

- Check signature status
  - should be done after the signature is verified
  - ensure the signature has not been revoked yet

- Once an intermediate signing key is validated
  - verify the signature of depending packets
  - recursively go back to the original data packet
Signature status checking

• Check if the signature has been revoked before expiration

• Verifier may retrieve signature status data
  – /<DataName>/[DataDigest]/[Timestamp]
  – content:
    • signature status: good, revoked
    • reasons (optional): revocation reasons

• Introduce a signature extension StatusChecking
  – ForwardingHint: where to forward the signature status interest
  – AuthorizedSigner: who can be trusted for signing signature status data

SignatureInfo:
  ...
  (Non)CriticalExtension:
  StatusChecking:
  ForwardingHint
  AuthorizedSigner
Thanks!