Trust Schema
Name-Based Trust Management

NDN Tutorial – ACM ICN 2015
September 30, 2015

Alex Afanasyev
University of California, Los Angeles
Goals

Recap of content-based authenticity approach of NDN

Learn insights on how NDN can make security usable

Learn how NDN apps can be secured today

What will be available soon
Overview

NDN architecture mandates signature
- Effectiveness of the mandate depends on the implementation
- If too complex, developers will shortcut
  - “temporarily” disable
  - use non-secure/fake signatures

Need a tool to make security usable
need automation
Data-Centric Security in NDN

- Data is named and is retrieved using name

- Name and content are bound together with a crypto signature

- Data packet includes a name of the public key to verify the signature
  - Key is also a data packet and retrievable by name

```
Name  Content  Signature  KeyLocator
```

```
Name  Content  Signature  KeyLocator
```

```
Name  Content  Signature  KeyLocator
```

```
Name  Content  Signature  KeyLocator
```

<table>
<thead>
<tr>
<th>Consumers</th>
<th>Producer</th>
</tr>
</thead>
<tbody>
<tr>
<td>retrieve data</td>
<td>sign</td>
</tr>
<tr>
<td>retrieve key</td>
<td>verify</td>
</tr>
</tbody>
</table>
Data Authentication

• To authenticate data, one needs a trust model
  • which keys are authorized to sign which data (trust rules)
  • one or more trusted keys
  • requires crypto properties

• Given trust model, anybody can verify data
  • applications
  • dedicated storage
  • routers

• **Trust model needs to be easily expressible**
  • help consumer to authenticate data
  • help producers to sign data
NDN Insight: Trust can be defined as a set of relationships between data and key names

Hierarchical trust relations

Cross-namespace trust relations

9/30/15
Desired Properties for Trust Policy Definition

- Clear definition of relationship rules
  - Use names and name patterns to define rules
    - data with /some/site prefix can be only signed with /some/site/key/<any-id>
    - keys /some/site/key/<any-id> can be only signed with /another/key/id=5
- Pre-configured trust anchors to bootstrap trust
  - /another/key/id=5
- Least privilege principle
  - Limited usage scope
  - Limited time-span
- Re-use of trust models between applications
  - Define, debug, and refine common trust models
- Make security easy to use
• Articles authored and signed by authors

• Authors are given permissions to publish on the blog by administrators

• Administrators are configured by blog configuration or other administrators
Web Blog Trust Relationships

All signs configured by

Blog Website

Admins

Authors

Articles

Authors

Admins

Blog Website

Articles

configured by

authorize

to publish

enable other

9/30/15

Generalized Rules for Name-Based Trust

Relationship between data and key names

- /a/blog/article/food/2015/3 <-> /a/blog/author/Alice/KEY/22
- /a/blog/article/drink/2014/9 <-> /a/blog/author/Zach/KEY/5

Generalizing relationship

- **blogPrefix** + “blog” + “article” + **category** + miscInfo <->
  - **blogPrefix** + “blog” + “**author**” + **name** + “KEY” + keyid

Use regular-based syntax to capture the relationship

- (<->)*<blog><article>[category]><><> <->
  - \1<blog><author>[user]<KEY>[id]
Web Blog: Trust Schema

Data Name

<table>
<thead>
<tr>
<th>Data Name</th>
<th>Key Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>article</td>
<td>(&lt;&gt;*&lt;blog&gt;&lt;article&gt;&lt;&gt;&lt;&gt;&lt;&gt;</td>
</tr>
<tr>
<td>author</td>
<td>(&lt;&gt;*&lt;blog&gt;&lt;author&gt;[user]&lt;KEY&gt;[id]</td>
</tr>
<tr>
<td>admin</td>
<td>(&lt;&gt;*&lt;blog&gt;&lt;admin&gt;[user]&lt;KEY&gt;[id]</td>
</tr>
</tbody>
</table>

Key Name

<table>
<thead>
<tr>
<th>Key Name</th>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>author(1)</td>
<td>/a/blog/article/food/2015/3</td>
</tr>
<tr>
<td>admin(1)</td>
<td>/a/blog/author/Alice/KEY/22</td>
</tr>
<tr>
<td>admin(1)</td>
<td>/a/blog/admin/Bob/KEY/5</td>
</tr>
<tr>
<td>root(1)</td>
<td>/a/blog/admin/Carl/KEY/37</td>
</tr>
</tbody>
</table>

Regexp-like pattern with grouping (group values accessible as \1, \2, \3 ...)

Name or other rule specializations

Different trust anchor for different blog website
Trust Rule Processing (1/3)

article

\[a/b/o/l\]

Content (article)

Signature

\[a/b/o/l/a\]

article must be signed with the key with name expanded from author("/a")

|[user]| -> accepts any user name (auth)
|-> generates use name (keygen)

|[id]| -> accepts any key id (auth)
|-> generates unique key id (keygen0)

author

\[a/b/o/l\]

(<>*)<blog><author>[user]<KEY>[id]

\[a/b/o/l\]
Trust Rule Processing (2/3)

/\a/blog/author/Yingdi/KEY/_v=5

Content (public key) 🍉

Signature

/\a/blog/admin/Alex/KEY

author

(<>*)<blog><author>[user]<KEY>[id]  \n
admin(\1)

/\a/blog/author/Yingdi/KEY/_v=5  \n
==>>  \1 = /\a

author key must be signed with the key with
name expanded from admin("/\a")

admin

(<>*)<blog><admin>[user]<KEY>[id]

<\a><blog><admin>[user]<KEY>[id]
Trust Rule Processing (3/3)

### admin

\[
/\text{a/blog/admin/Alex/KEY/_v=1} \quad \Rightarrow \quad 1 = /\text{a}
\]

author key must be signed with the key with name expanded from `admin(“/a”)`

OR

key expanded from `root(“/a”)`

### admin

\[
 любимый
\]

Content (public key)

Signature

\[
/\text{a/blog/admin/Yingdi/KEY}
\]

### admin

\[
 любимый
\]

Content (public key)

Signature

\[
/\text{a/blog/admin/Lixia/KEY/_v=1}
\]

### root

\[
 любимый
\]

Content (public key)

Signature

\[
/\text{a/blog/KEY}
\]
NDN Link-State Routing (NLSR)

Network (NDN Testbed)

Site (UCLA)

Site (UM)

Site (UA)

NDN Testbed Root → Site → Operator → Router → NLSR instance → LSA version

NDN Testbed Root

Site

Operator

Router

NLSR instance

LSA version

NLSR instance

Link LSA

Name LSA

Coordinate LSA
NLSR Trust Relationships

NDN Testbed Root
/ndn/KEY/ksk-1....

Site key
/ndn/[site-name]/KEY/...

Site key
/ndn/[site-name]/USR/[user]KEY/...

Router key
/ndn/[site-name]/RTR/[hostname]KEY/...

NLSR instance key
/ndn/[site-name]/RTR/[hostname]/NLSR/KEY/...

LSA key
/ndn/NLSR/LSA/[site-name]/RTR/[hostname]/NLSR/[type]/[id]
## NLSR: Trust Schema

<table>
<thead>
<tr>
<th>Data Name</th>
<th>Key Name</th>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSA</td>
<td>instance(1, 2, 3)</td>
<td>/ndn/NLSR/LSA/edu/ucla/RTR/spurs/n/1</td>
</tr>
<tr>
<td>instance</td>
<td>router(1, 2)</td>
<td>/ndn/edu/ucla/RTR/spurs/NLSR/KEY/22</td>
</tr>
<tr>
<td>router</td>
<td>operator(1)</td>
<td>/ndn/edu/ucla/RTR/spurs/KEY/44</td>
</tr>
<tr>
<td>operator</td>
<td>site(1)</td>
<td>/ndn/edu/ucla/USR/alex/KEY/2</td>
</tr>
<tr>
<td>site</td>
<td>root</td>
<td>/ndn/edu/ucla/KEY/5</td>
</tr>
</tbody>
</table>

### Key Name

- **root**: `<ndn><KEY>[id]`

### Key

- `/ndn/KEY/ksk-1.. (MIIBID...)`

**NDN Testbed Root**
Tutorial Chat App

icn2015.ndncert.named-data.net

Approve request for ICN user key

Check email validity

Requests ICN user key

ICN user

Create new channel

Join Channel

Request for channel owner key

Request for channel user key

ICN chat channel service

Record taken channel names

Issue channel owner key

Issue channel user key

Chat channel owner

Chat channel user

Post chat message

Run ChronoSync to synchronize knowledge about messages
ICN root key
/ndn/org/icn/KEY/...

ICN user key
/ndn/org/icn/USER/[email]/KEY/...

ICN chat channel owner key
/ndn/org/icn/CHAT/CHANNEL/[name]/KEY/...

Request to join an existing channel
/ndn/org/icn/CHAT/CHANNEL/[name]/JOIN-REQUEST/[PUBLIC-KEY]/...

NFD prefix registration
/localhost/nfd/nb/register/...

ICN chat channel user key
/ndn/org/icn/CHAT/CHANNEL/[name]/USER/[email]/KEY/...

Chat message
/ndn/org/icn/USER/[email]/CHAT/CHANNEL/[name]/SESSION/[id]/[seq]

ChronoSync replies
/ndn/multicast/CHAT/CHANNEL/[name]/[digest]
## Trust Schema

<table>
<thead>
<tr>
<th>Data Name</th>
<th>Key Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>create_channel</td>
<td>user(\1, \2)</td>
</tr>
<tr>
<td>join_channel</td>
<td>user(\1, \2)</td>
</tr>
<tr>
<td>chat_data</td>
<td>channel_user(\1, \2)</td>
</tr>
<tr>
<td>sync_reply</td>
<td>channel_user(\1, \2)</td>
</tr>
<tr>
<td>channel_user</td>
<td>channel_owner(\1, \2)</td>
</tr>
<tr>
<td>channel_owner</td>
<td>root(\1)</td>
</tr>
<tr>
<td>user</td>
<td>root(\1)</td>
</tr>
</tbody>
</table>

### Key Name

<table>
<thead>
<tr>
<th>Key Name</th>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>root</td>
<td>Mb43ha38as3Hbb...</td>
</tr>
</tbody>
</table>
Making Trust Schema Universal Tool for Trust

Captures data/key name relationships using generalizations and patterns
  • formally describes and defines trust model
  • enforces trust model in automatic way
    • both authentication and signing paths

Representable in a data packet
  • can be retrieved and executed by any NDN entity
  • can be (recursively) authenticated using higher-level schemas

Trust schema also defines security design pattern
  • regulate the behavior of applications
    • an operating system can define a trust schema to authenticate the trust schema of applications
    • only install and execute apps with authenticated trust schema
Trust Schema Implementation Status

- old schema (ValidatorConf)
- new schema implementation in the upcoming release

NDN-CCL: [http://named-data.net/codebase/platform/ndn-ccl/](http://named-data.net/codebase/platform/ndn-ccl/)
- NDN-CPP, NDN-JS, PyNDN, jNDN

Trust schema powers data and interest authentication in
- NFD: NDN Forwarding
- NLSR: NDN Link State Routing Protocol
- Repo-ng: NDN Data Repository
- ChronoChat: a chat application over NDN
- NDNS: NDN Domain Name System

Works! Even better implementations coming really soon
Trust Schema Specification (New Format)

Rule
• Restriction on a packet name and its signing key name

Anchor
• A pre-authenticated public key (a data packet carrying the public key)

Crypto (Signature Requirements)
• Cryptographic requirements on packet signature: which public key algorithm to use, which hashing algorithm to use, and what is the minimum required signature strength
Rule

id
• a unique identifier of the rule in the trust schema that can be used to link rules as part of signer "function." The identifier must start with a letter, and can only contain letters, digits, and underscores. The identifier is case-sensitive.

name
• name pattern of the packet in terms of NDN regular expression

type
• data (default) and interest

signer
• one or more invocations of rules or trust anchors, separated by |
### Anchor

**id**
- identifier for the anchor that can be used to link an anchor to a rule as a signer "function". The identifier must start with a letter, and can only contain letters, digits, and underscores.

**name**
- name pattern of the packet in terms of NDN regular expression

**file, raw, or dir**
- filename, base64 content, or directory containing pre-authenticated public key certificate

```
anchor
{
  id root
  name (<>*<blog><KEY>[id]
  file blog-root.cert
}
```

```
anchor
{
  id another-root
  name <KEY>[id]
  raw
  "Bv0DGwdG...amHFvHIMDw=="
}
```
Crypto (Signature Requirements)

hash
• one or more allowed hash algorithms, separated by |

signing
• one or more allowed signing algorithms, separated by |
• rsa (RSA signature algorithm), ecdsa (ECDSA signature algorithm)

key-strength
• minimum crypto strength of a key (in terms of symmetric key bits)
Current Status / Future Work
Trust Schema (Old Format/ValidatorConf)

rule
{
  id "Simple Rule"
  for data
  filter
  {
    type name
    name /localhost/example
    relation is-prefix-of
  }
  checker
  {
    type customized
    sig-type rsa-sha256
    key-locator
    {
      type name
      name /ndn/edu/ucla/Yingdi/KEY/1
      relation equal
    }
  }
}

rule
{
  id simple_rule
  name <localhost><example><>*
  signer /ndn/edu/ucla/Yingdi/KEY/1
}

+ crypto
{
  hash sha-256
  signing rsa | ecdsa
  key-strength 112
}
A Few Details and Future Plans in NDN Security

The currently used format for NDN public key

<table>
<thead>
<tr>
<th>Name</th>
<th>MetaInfo</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ContentType: KEY(2)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SignatureInfo</th>
</tr>
</thead>
<tbody>
<tr>
<td>SignatureType:</td>
</tr>
<tr>
<td>KeyLocator:</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SignatureValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

The new format in the upcoming release

<table>
<thead>
<tr>
<th>Name</th>
<th>MetaInfo</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ContentType: KEY(2)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Public Key</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SignatureInfo</th>
</tr>
</thead>
<tbody>
<tr>
<td>SignatureType:</td>
</tr>
<tr>
<td>KeyLocator:</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SignatureValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
</tr>
</tbody>
</table>
Take out points

• Trust schema becoming a universal tool for trust
  • captures trust models in a concise and generalized way
  • contained in data packet(s) and recursively secured with the trust schema
  • is a new design pattern for security