http://named-data.net/icn2015-tutorial

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 be available soon

Overview

NDN architecture mandates signature

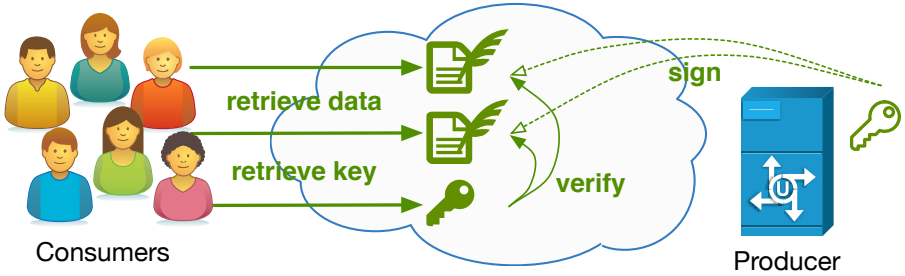
- Effectiveness of the mandate depends on the implementation
- If too complex, developers will shortcut
  - “temporarily” disable
  - use non-secure/fake signatures

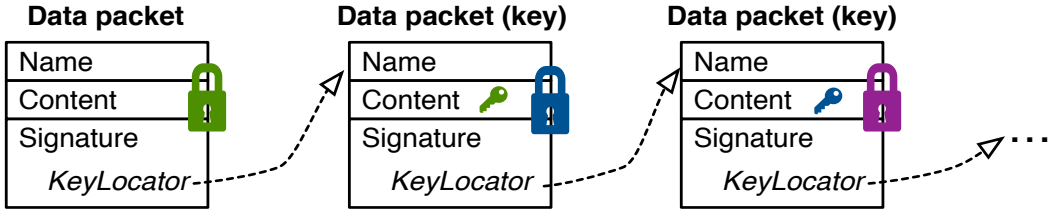
Need a tool to make security usable

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





Data Authentication

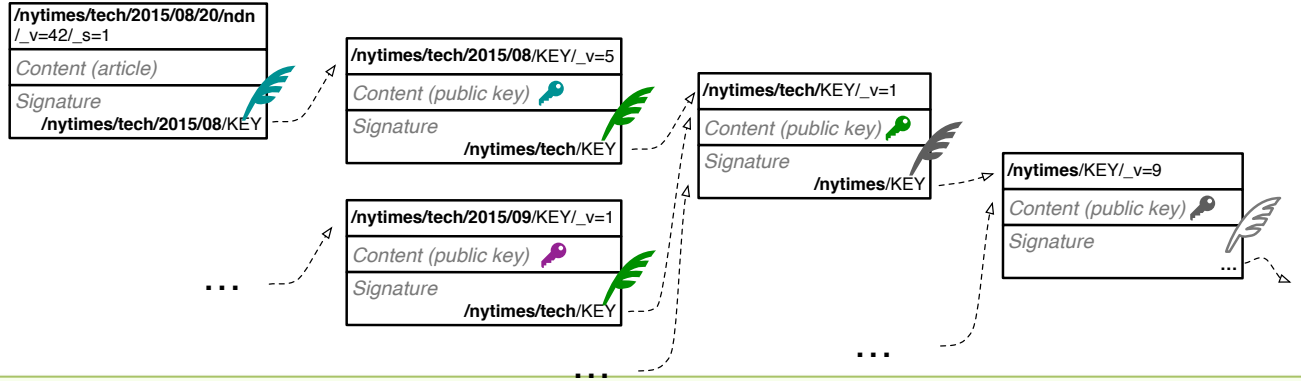
- To authenticate data, one needs a trust model
  - which keys are authorized to sign which data (trust rules)
  - one ore more trusted keys
  - requires crypto properties

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

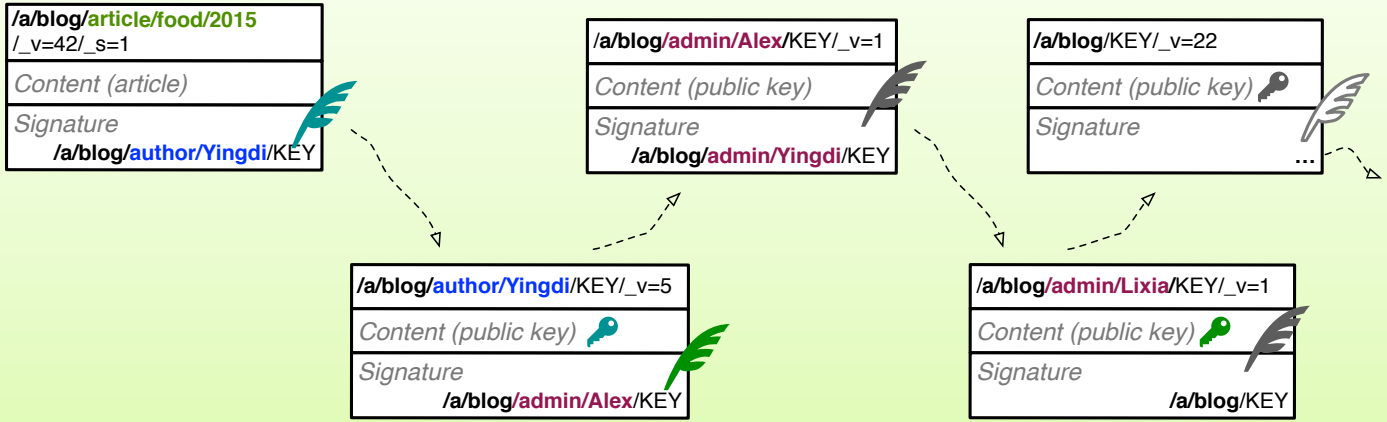
- **Trust model needs to be easily expressible**
  - help consumer to authenticate data
  - help producers to sign data

NDN Insight: Trust can be defined as a set of relationships between data and key names

Hierarchical trust relations



Cross-namespace trust relations

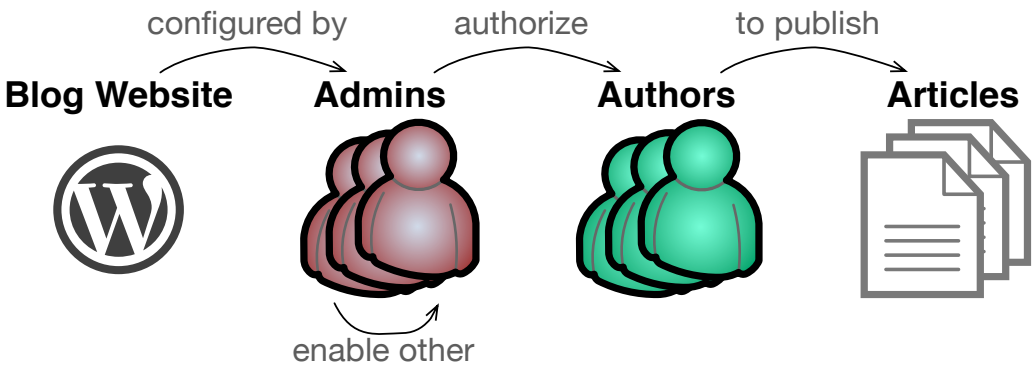


Desired Properties for Trust Policy Definition

- 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
    - `/anoth`
- Least privileg
  - 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

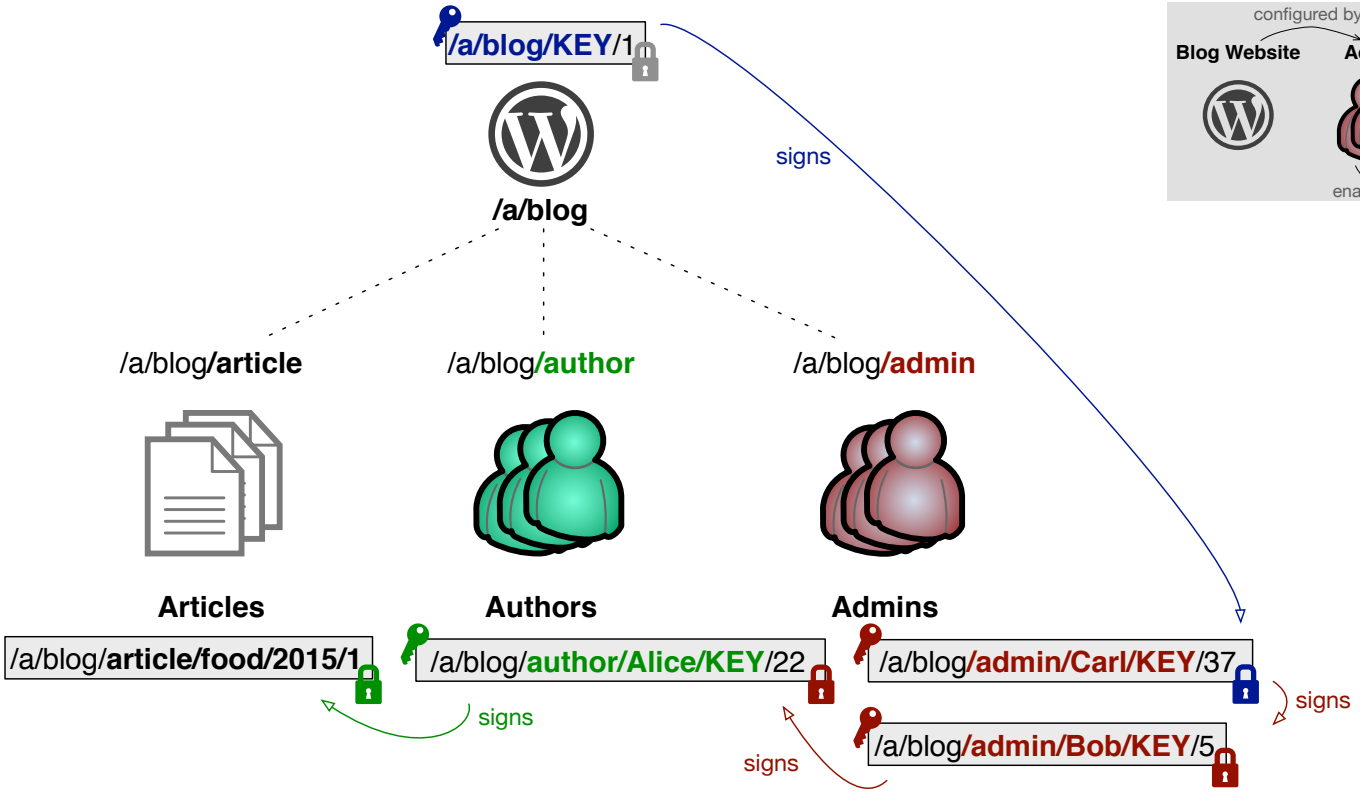
Trust Schema to Schematize and Generalizing Trust

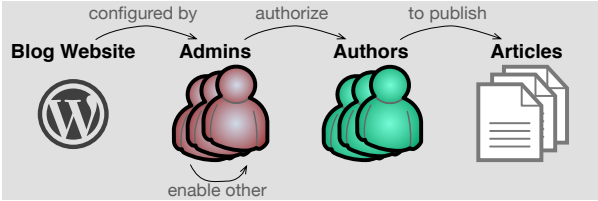
Web Blog



- 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





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

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

Name or other rule specializations

	Data Name	Key Name	
article	(<>*)<blog><article><><><>	**author(\1)**	
author	**(<>*)**<blog><author>[user]<KEY>[id]	**admin(\1)**	
admin	**(<>*)**<blog><admin>[user]<KEY>[id]	**admin(\1)**	**root(\1)**

/a/blog/article/food/2015/3

/a/blog/author/Alice/KEY/22

/a/blog/admin/Bob/KEY/5

/a/blog/admin/Carl/KEY/37

	Key Name	Key
root	**(<>*)**<blog><KEY>[id]	/a/blog/KEY/1 (0x30 0x82 ...)

Different trust anchor for different blog website

Trust Rule Processing (1/3)

/a/blog/article/food/2015
/_v=42/_s=1

Content (article)

Signature
/a/blog/author/Yingdi/KEY

article	**(<>*)**<blog><article><><><>	**author(\1)**

/a/blog/article/food/2015/3 =>> \1 = /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	(<>*)<blog><author>[user]<KEY>[id]	**<a>**<blog><author>[user]<KEY>[id]

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] **admin(\1)**

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

/a/blog/admin/Alex/KEY/_v=1
Content (public key)
Signature /a/blog/admin/Yingdi/KEY

/a/blog/admin/Lixia/KEY/_v=1
Content (public key)
Signature /a/blog/KEY

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

/a/blog/admin/Alex/KEY/_v=1 =>> \1 = /a

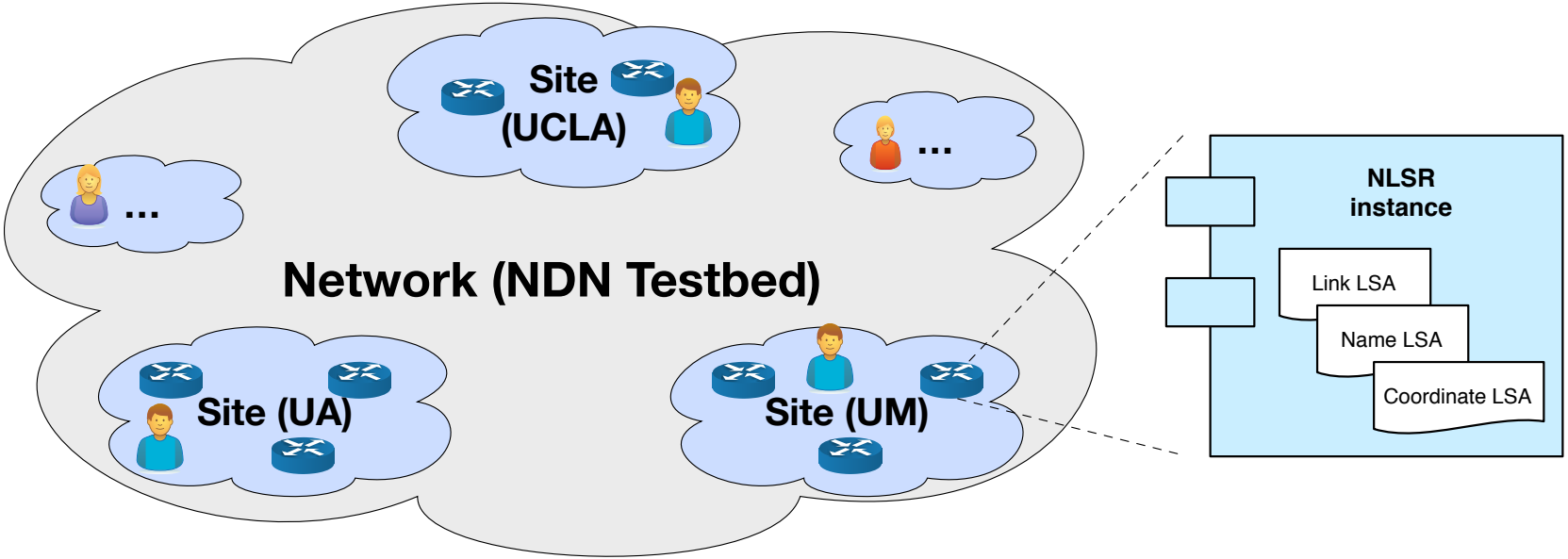
author key must be signed with the key with name expanded from **admin("/a")**

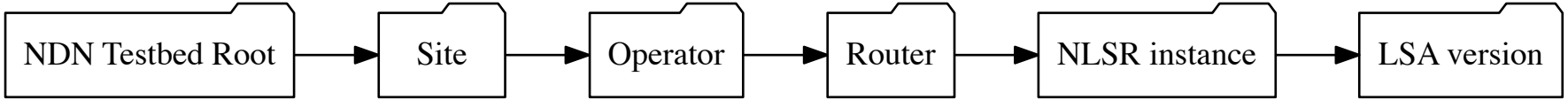
OR

key expanded from **root("/a")**

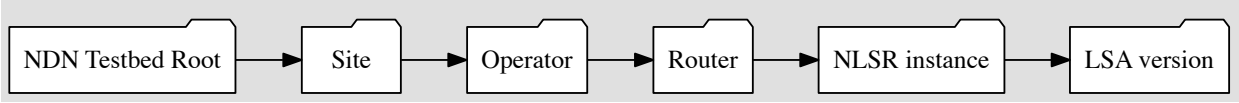
admin	(<>*)<blog><admin>[user]<KEY>[id]	**<a>**<blog><admin>[user]<KEY>[id]
root	(<>*)<blog><KEY>[id]	**<a>**<blog><KEY>[id]

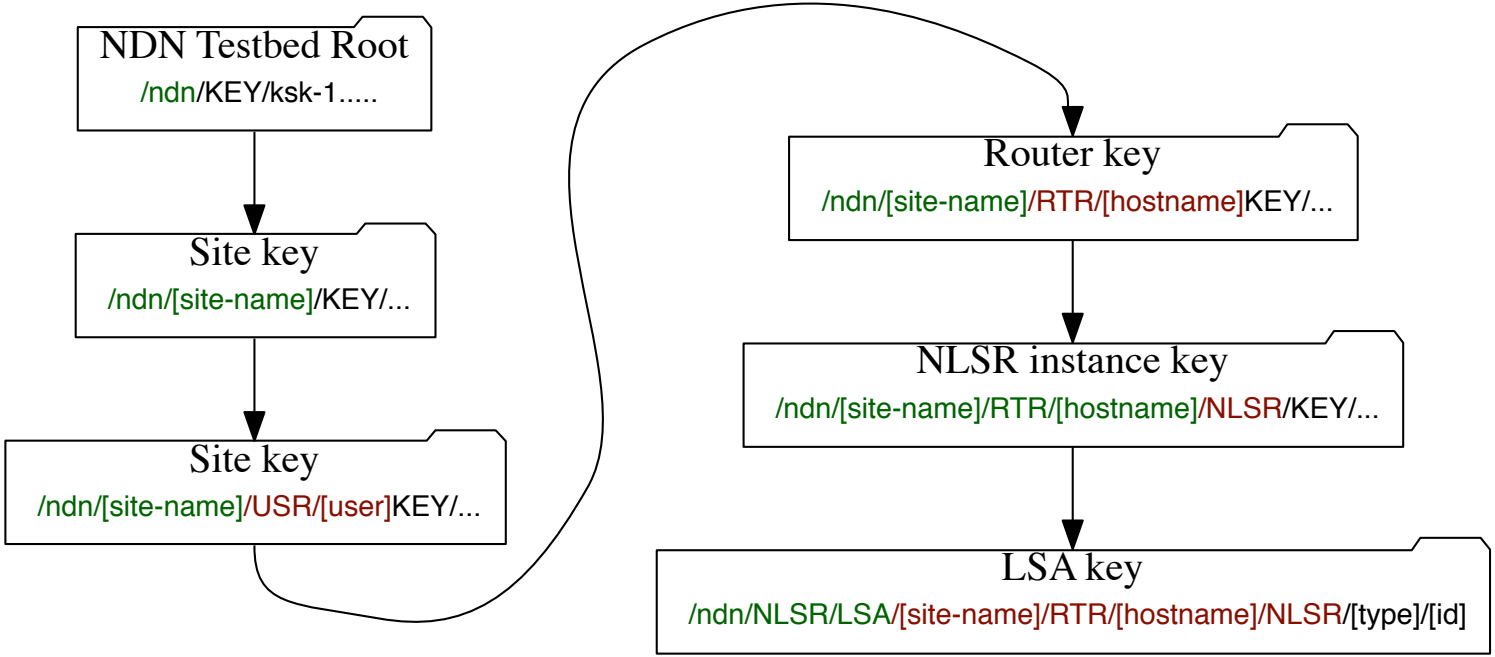
NDN Link-State Routing (NLSR)





NLSR Trust Relationships





NLSR: Trust Schema

	Data Name	Key Name
LSA	(<>*)<NLSR><LSA>(<>*)<RTR>(<>)<><>	instance(\1, \2, \3)
instance	(<>*)<RTR>(<>)<NLSR><KEY>[id]	router(\1, \2)
router	(<>*)<RTR>(<>)<KEY>[id]	operator(\1)
operator	(<>*)<USR>[user]<KEY>[id]	site(\1)
site	(<>*)/KEY/[id]	root

/ndn/NLSR/LSA/**edu/ucla**/RTR/**spurs**/n/1

/ndn/**edu/ucla**/RTR/**spurs**/NLSR/KEY/22

/ndn/**edu/ucla**/RTR/**spurs**/KEY/44

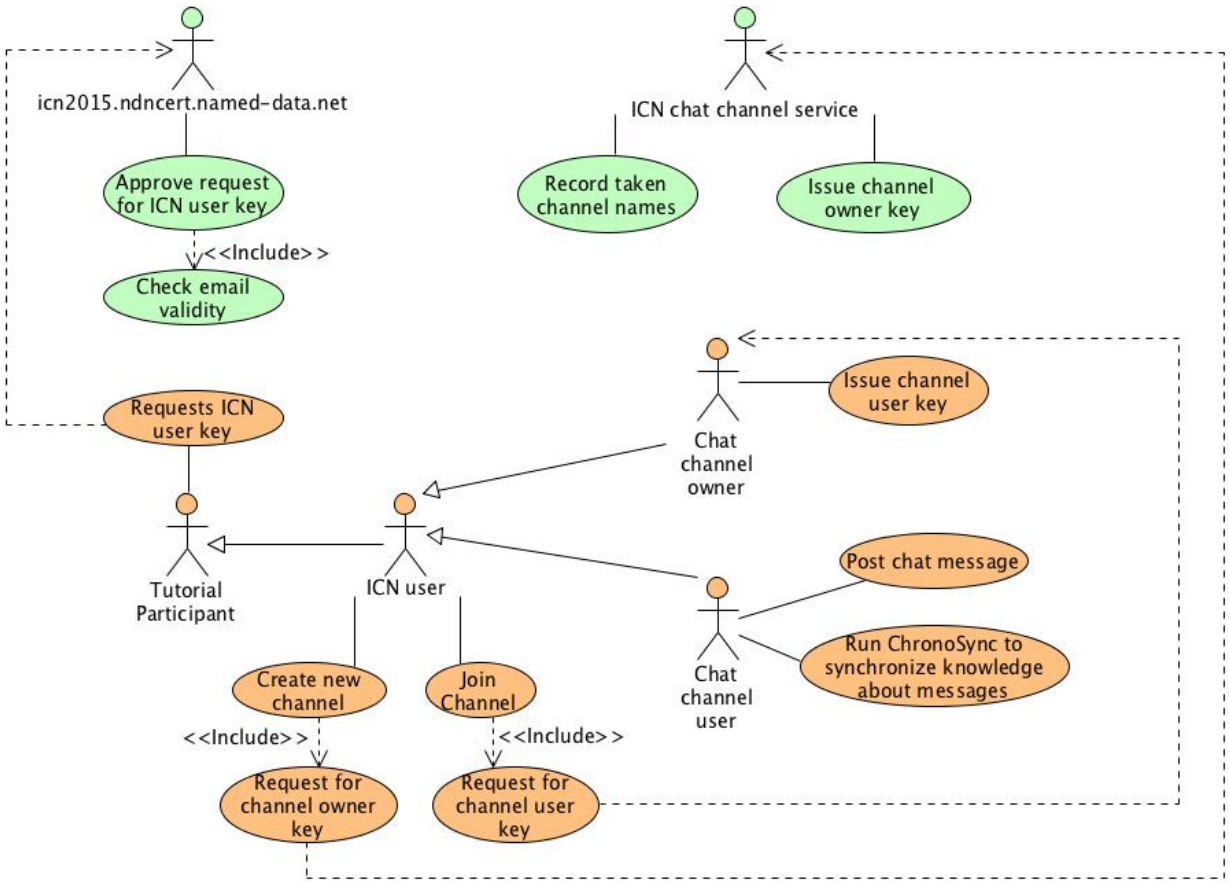
/ndn/**edu/ucla**/USR/**alex**/KEY/2

/ndn/**edu/ucla**/KEY/5

	Key Name	Key
root	**<ndn>**<KEY>[id]	/ndn/KEY/ksk-1.. (MIIBID...)

NDN Testbed Root

Tutorial Chat App



Tutorial App Trust Relationships

Namespaces

ICN root key
/ndn/org/icn/KEY/...

signs

signs

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

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

signs

signs

signs

signs

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

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

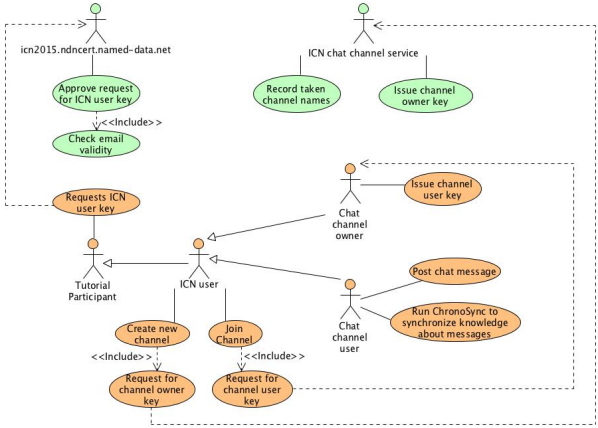
ICN chat channel user key
/ndn/org/icn/CHAT/CHANNEL/[name]/
USER/[email]/KEY/...

signs

signs

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

ChronoSync replies
/ndn/multicast/CHAT/CHANNEL/[name]/[digest]



Trust Schema

	Data Name	Key Name
create_channel	(<>*)<CHANNEL-CREATION-REQUEST><>	user(\1, <>)
join_channel	(<>*)<CHAT><CHANNEL><><JOIN-REQUEST><>	user(\1, <>)
chat_data	(<>*)<USER>(<>)<CHAT><CHANNEL><><SESSION><><>	channel_user(\1, \2)
sync_reply	<ndn><mulitcast><CHAT><CHANNEL><><>	channel_user(<ndn><org><icn>, <>)
channel_user	(<>*)<CHAT><CHANNEL>(<>)<USER>(<>)<><KEY>	channel_owner(\1, \2)
channel_owner	(<>*)<CHAT><CHANNEL>(<>)<><KEY>	**root**(\1)
user	(<>*)<USER>(<>)<><KEY>	**root**(\1)

	Key Name	Key
root	(<ndn><org><icn>)<><KEY>	Mb43ha38as3Hbb…

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

ndn-cxx: http://www.github.com/named-data/ndn-cxx

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

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 |

```
rule
{
  id article
  name
(<>*)<blog><article><><><>
  signer author($1)
}
```

```
rule
{
  id key
  name (<>*)(<>)<KEY>[id]<ID-
CERT>[version]
  signer key($1,null)|root()
}
```

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

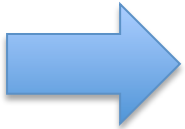
- minimum crypto strength of a key (in terms of symmetric key bits)
- http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57_part1_rev3_general.pdf

```
crypto
{
  hash sha-256
  signing rsa | ecdsa
  key-strength 112
}
```

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

```
Name
MetaInfo
  ContentType:            KEY(2)
Content
  x.509 DER
  Validity
  Name
  SubjectPublicKeyInfo
  Extensions ...
SignatureInfo
  SignatureType:          ...
  KeyLocator:             ...
  ...
SignatureValue
```

The new format in the upcoming release

```
Name
MetaInfo
  ContentType:            KEY(2)
Content
  Public Key
SignatureInfo
  SignatureType:          ...
  KeyLocator:             ...
  ValidityPeriod: ...
  ...
SignatureValue
```

Q A

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