Name-Based Access Control

NDN Tutorial – ACM ICN 2015
September 30, 2015

Yingdi Yu
University of California, Los Angeles
What we have so far...

- Facilitating data retrieval over the network
  - Synchronization as the new transport paradigm in NDN

- Making data verifiable independently from where data is retrieved
  - Trust schema as a new mechanism to specify application-level trust model

- Can we also have location-independent content confidentiality?
What you will get next…

• Content-based confidentiality
  • the other part of content-based security
  • confidentiality stays with content, no dependency on the content delivery system

• End-to-end confidentiality
  • the “end” of application-level communication
    • more general, including multi-party communication
  • not the “end” of a connection

• Differential confidentiality
  • granting data access at fine granularities

• Multi-party access control system
  • coordinate access control among multiple data producers and consumers
Application Scenario

• Alice collects her health data through a fit application

• Alice wants to share different health data with different people
  • share her daily activity data with Bob
  • occasionally share her step data with Cathy
  • share her blood sugar data with David after each meal

• No one except content owner (Alice), producers (sensors), and authorized consumers (e.g., Bob) can see the data
Content-Based Access Control

- Data owner (Alice) enforces access control on data directly
  - independent from data container (storage, channels)

- Write access
  - produce data on behalf of owner
  - issue signing certificate

- Read access
  - encrypt data
  - distribute decryption key

- Data container is a pure storage
  - the only always-online entity
Container-Based Access Control

- Data owner relies on a container (e.g., application server) to enforce access control

- Limitations
  - data owner has to trust the container
  - container must understand ACL semantics
  - container must be able to authenticate entities
  - content must be delivered over a secure channel
Privilege: Data Namespace

• Producers can produce data under its own namespace
  • blood sugar sensor:
    • /alice/health/samples/medical/bloodsugar
  • activity sensor
    • /alice/health/samples/activity

• Consumers can only read data under the authorized namespace
  • data owner may enforce further restriction
    • e.g., data produced during certain time periods, or at certain locations
Simple Encryption-Based Read Access

- **Producer side**
  - creates a content key
    - e.g., /alice/health/samples/activity/location/C-KEY
  - encrypt data using content key
  - encrypt content key using the encryption key of authorized consumers
    - e.g., /bob/health/access/E-KEY

- **Consumer side**
  - retrieve encrypted data
  - retrieve encrypted content key
  - decrypt content key
    - e.g., /bob/health/access/D-KEY
  - decrypt data
  - as long as data is available
Encrypted Content Format

- Data packet must carry enough information for authorized consumers to decrypt content

- Experiment as application semantics
  - content encoding (not a part of architecture yet)

- EncryptedContent TLV contains three sub-TLVs:
  - EncryptionAlgorithm
    - may also algorithm-specific fields, e.g., Initial Vector
  - DecryptionKeyName
    - used by the consumer to retrieve the decryption key
  - EncryptedContent

- When a data has more than one encrypted copies
  - each encrypted copy is an independent data packet
  - naming convention: /<content_name>/FOR/<decrypt_key_name>
Decryption Chain

- Consumer Bob extracts DecryptionKeyName
  - `/alice/health/samples/activity/location/C-KEY`
- C-KEY is encrypted, construct an interest by appending its own name
  - `/alice/health/samples/activity/location/C-KEY/FOR/bob/health/access`
- The interest will bring back a content key encrypted using one of Bob’s public key
Delegated Data Production

- In some scenarios, data owner may delegate the data production to others
  - e.g., Alice gives her activity sensors to produce data on her behalf.
  - delegated producer may not know the authorized consumers

- Data owner should be able to direct the content encryption at the producer side
  - create a content key
  - publish the key encrypted with both producer and consumer keys
  - producer retrieves content key and encrypt content
Limitations in Distributed Production

• Producer cannot produce data without fetching the content key
• Data owner has to be online all the time to keep producing content key
• Data owner has to be aware of all the potential producers
  • because content key must be encrypted using producer’s public key

• Can we decouple data production from access control?
• Producer is free to encrypt content
• Data owner is free to grant access
Name-Based Read Access Control

- Data owner create a pair of public/private key instead of a content key
  - the name of public key defines a read access scope
  - any producer that produces data in the access scope should encrypt data through the public key (group encryption key)
  - any consumer that obtains the private key (group decryption key) can read the data in access scope

- Producer creates its own content key
  - encrypt data using its own content key
  - encrypt content key with appropriate group encryption key
Group Key Namespace

- Only data owner is allowed to create group key

- Group keys are named under a separate namespace
  - `/alice/health/read` vs `/alice/health/samples`
  - so that data producers are not allowed to create group keys

- Group key namespace replicate data namespace
  - `/alice/health/read/activity`
  - `/alice/health/read/medical/bloodsugar`
Group Key Naming Convention

• Key name defines the access scope

• Group encryption key
  • /<data_prefix>/E-KEY/<additional_restriction>
  • data_prefix: describe the prefix of data to encrypt
  • additional_restriction: further classify encrypted data
    • e.g., producing timestamp, geo-location
  • A producer can tell which content key should be encrypted using the group encryption key according to the key name

• Group decryption key
  • /<data_prefix>/D-KEY/<additional_restriction>/FOR/<consumer_key_name>
  • consumer_key_name: the name of consumer key that can decrypt the group key
  • uniquely identify one copy of encrypted group key
How to retrieve group encryption key?

• Case 1: continuous data production
  • e.g., health sensors produce data all the time
  • data owner needs to generate group encryption keys that can cover all the time
    • data owner can encode time information into additional_restriction portion of key name
      • `/<data_prefix>/E-KEY/<start_ts>/<end_ts>`
      • e.g., `/alice/health/read/activity/E-KEY/20150930160000/20150930180000`
    • a producer once retrieve an encryption key, it knows the starting timestamp of next encryption key thus it can send interest
      • `/alice/health/read/activity/E-KEY/20150930180000`
      • and retrieve next encryption key,
        • e.g., `/alice/health/read/activity/E-KEY/20150930180000/20151001000000`

• Case 2: arbitrary additional restriction
  • data owner create a sync group for encryption keys, and all producers join the sync group
Data Type Granularity

- Data owner may also specify a variety of data_prefixes for different access granularity
  - Alice may share her activity data with Bob, but only share her step data with Cathy
    - Bob can access both location and step data, while Cathy can access step data only
  - two group keys:
    - /alice/health/read/activity/D-KEY \rightarrow Bob
    - /alice/health/read/activity/step/D-KEY \rightarrow Cathy

- Producer also need to track encryption keys with different granularities
  - Alice’s activity sensor needs to track
    - /alice/health/read/activity/step/E-KEY/
    - /alice/health/read/activity/E-KEY/
    - /alice/health/read/E-KEY/
  - when sync is used, encryption keys of the same granularity forms a data set to synchronize between data owner and producers

9/30/15
Example: Data Owner

• Alice wants to share her activity data during 4pm-6pm Sep. 30, 2015 with her friend Bob

• Create an encryption key:
  • /alice/health/read/activity/E-KEY/20150930160000/20150930180000
  • signed by Alice’s key, so that her activity sensor can authenticate the E-KEY.

• Encrypt the decryption key using Bob’s public key
  • /alice/health/read/activity/D-KEY/20150930160000/20150930180000/FOR/bob/health/access
  • signed by Alice’s key, so that Bob can authenticate the D-KEY.
Example: Producer

- Alice’s activity sensor produce step data at time 17:00 Sep. 30, 2015
  - /alice/health/samples/activity/steps/201509301700

- Create a content key
  - /alice/health/samples/activity/steps/C-KEY/20150930170000
  - encrypt step data using the content key
  - signed by sensor’s key

- Time of the step data falls between the interval of 4pm-6pm
  - encrypt the content key using the group encryption key
  - /alice/health/samples/activity/steps/C-KEY/20150930170000/FOR/alice/health/samples/activity/D-KEY/20150930160000/20150930180000/
  - signed by sensor’s key
Example: Consumer

- Bob retrieve Alice’s step data: /alice/health/samples/activity/steps/201509301700
  - data contains the decryption key name:
    • /alice/health/samples/activity/steps/C-KEY/20150930170000

- Bob knows Alice share the activity data with him: /alice/health/read/activity
  - send an interest for the content key encrypted using the group key for
    • /alice/health/samples/activity/steps/C-KEY/20150930170000/FOR/alice/health/samples/activity
  - Bob does not have to specify the full name of the group decryption key

- When encrypted content key is retrieved
  - Bob learns the exact name of the group decryption key from the packet payload:
    • /alice/health/read/activity/D-KEY/20150930160000/20150930180000
  - send another interest for the group decryption key encrypted using Bob’s key
    • /alice/health/read/activity/D-KEY/20150930160000/20150930180000/FOR/bob/health/access

- Decrypt the group decryption key using its own private key, and then decrypt the content key and sync data
Post-fact Access Granting

• It is necessary to grant access to data that has been produced long time ago

• Data owner should always retain a copy of content key
  • it is resource consuming to store a copy at the data owner side
  • data owner only needs to create a super group key
    • /alice/health/read/E-KEY
  • data owner keep the decryption key to itself
  • all producers under the data namespace will encrypt every content key using the encryption key
  • when a data owner needs to grant the access later
    • retrieve the encrypted content key
    • re-encrypt the content key with the authorized consumer’s encryption key.
What to expect in next step?

• An application library will be available in next NDN platform release

• Convert key exchange between data owner and producers to identity-based encryption (or attribute-based encryption)

• Enable forward secrecy: decouple consumer private key with content key
  • minimize the damage when a private key is compromised later

• Revoke access that has been granted & prevent unauthorized access granting
  • controlled functional encryption

• Future work
  • Name privacy
  • Read auditing
  • Secure multi-party computing
Conclusion

• Content-based confidentiality
  • true “end-to-end” confidentiality

• Differential confidentiality
  • leveraging hierarchical namespace
  • multi-dimensional access control

• Effective access control
  • separate access control apart from data production

• Still many open questions
Content-Based Write Access Control

- **Write access**
  - endorse one to produce data on behalf of data owner
    - a channel owner may allow all participants to publish sync data on behalf of the channel owner
  - data modification is equivalent to producing a new version of data

- **Enforce write access**
  - data owner issues certificates to producers
  - data consumer authenticates data
    - discard data produced by non-authorized producers

- **Granularity**
  - through the name of producer certificate
Write Access Example

• The owner a chat channel “demo” owns the channel’s sync data namespace
  • /ndn/multicast/CHAT/CHANNEL/demo

• The owner may issue each participant a channel specific certificates (channel user certificate)

• With trust schema, consumers will accept sync data produced by user with a valid channel user certificate

• Any other chat channel related data (e.g., channel description) will not be accepted if the producer is a channel user