Data Confidentiality and Access Control
Name-based Access Control

Zhiyi Zhang (UCLA)
Today’s Confidentiality and Access Control

◊ Traditional connection-based confidentiality
  o End-to-End Confidentiality? (e.g., CDN, Middlebox)
  o Multi-party confidentiality and access control?

◊ Encrypt Content Directly over TCP/IP architecture
  o Inefficient Key Distribution (where to fetch those keys?)
  o Content Multicast
Confidentiality and Access Control over NDN

◊ Built-in Data Authentication and Integrity
  o Integrity helps the confidentiality (e.g., MITM)

◊ Automating key distribution with naming conventions
  o Extract decryption key name from content and Data name
  o Fetch decryption key directly using name

◊ Content Delivery and In-network Cache
Name-Based Access Control (NAC) Concepts

Access Manager

Encryptor

Decryptor

(Data Owner) Entity that control access to the data associated with the namespace

Grants access

(Producer) Entity that encrypts data based on namespace association

(Consumer) Entity that decrypts data based on namespace association

Defines encryption policy
Name-Based Access Control (NAC) Concepts

**Access Controller**—Android Phone
- Creates a list of encryption/decryption key pairs
- Controls which encryption keys are used to encrypt which namespace
- Control whom to distribute the corresponding decryption keys

**Producers (Encryptors)**—Camera
- Fetch the right encryption keys to encrypt data (2 steps)

**Consumers (Decryptor)**—Television
- Fetch the right decryption keys to decrypt data (3 steps)
NAC Process

Encryption Key (KEK)

Decryption Key (KDK)

Content Key (C-KEY)

Data

Android Phone (data owner)

Decryption Key (KDK)

Content Key (C-KEY)

Data

Public Key

Camera (data producer)

Television (data consumer)
An example of NAC’s naming convention

Control the access to the smart home camera data

◊ The owner/controller of the access control system
  o /home/controller

◊ The authorized consumer:
  o /home/LivingRoom/Television-01

◊ The dataset that is being controlled:
  o /home/LivingRoom/Camera-01/FrontView

◊ The producer:
  o /home/LivingRoom/Camera-01
Access Manager (aka Data Owner)

• Encryption policy using public key (KEK)
  /home/controller/NAC
  /home/LivingRoom/Camera-01/FrontView/KEK/<key-id>

• Authorizes decryptors by publishing encrypted version of private key (KDK)
  /home/controller/NAC
  /home/LivingRoom/Camera-01/FrontView/KDK/<key-id>
  /ENCRYPTED-BY
  /home/LivingRoom/Television/KEY/<key-id>
Encryptor (aka Producer)

- Encrypts input data using CK, returns encrypted content
- Exact name of the corresponding CK data is embedded in the encrypted content

From Access Manager / provisioned or dedicated data owner storage

- Fetches and stores KEK for the configured with access prefix
- Interest ->
  /home/controller/NAC
  /home/LivingRoom/Camera-01/FrontView/KEK

- Generates (re-generates) symmetric Content Key (CK)
- Publishes CK data under configured namespace, encrypted by KEK

Data:
- /home/LivingRoom/Camera-01/CK/<key-id>
- /ENCRYPTED-BY
  /home/controller/NAC/KEK/<key-id>
Decryptor (aka Consumer)

- Fetches CK data for the name extracted from input encrypted payload

Interest->
/home/LivingRoom/Camera-01/CK/<key-id>

- Fetches KDK, name extracted from CK name + own configured access key name

Interest->
/home/controller/NAC
/home/LivingRoom/Camera-01/FrontView/KDK/<key-id>

/ENCRYPTED-BY
/home/LivingRoom/Television/KEY/<key-id>

From Encryptor / from same place as data

- Fetch the encrypted Content Data
- Get the name of the corresponding CK: CK name is embedded in the encrypted content

From Access Manager / provisioned or dedicated data owner storage
Fine Granularity: Play with Names

Possible Granularity:

◊ /home/LivingRoom/Camera-01, /home/BedRoom/Camera-02

◊ /home/LivingRoom/Camera-01/FrontView, /home/LivingRoom/Camera-01/BackView

◊ /home/LivingRoom/Camera-01/FrontView/8AM-10AM, /home/LivingRoom/Camera-01/FrontView/10AM-12PM

◊ …
NAC Library API Highlights

```
#include "access-manager.hpp"
...

AccessManager accessManager(identity, granularity, ...);
accessManager.addMember(authorizedCert1);
accessManager.addMember(authorizedCert2);
```

Encryptor encryptor(accessPolicyName, ckName, ...);

Data data(dataName);
data.setFreshnessPeriod(10_s);

auto content = encryptor.encrypt(data, dataSize);
data.etContent(content.wireEncode());

keyChain.sign(data);

Decryptor decryptor(identity, ...);
decryptor.decrypt(data.getContent().blockFromValue(),
  [=] (ConstBufferPtr content) {
    ...
  },
  [=] (const ErrorCode&, const std::string& error) {
    std::cerr << "Cannot decrypt data: " << error << std::endl;
  });
Scalability?

◊ In NAC, the complexity of key generation is \( O(m) \), and complexity of key distribution is \( O(m \times n) \) where

- \( m \) is the number of granularities
- \( n \) is the number of consumers

◊ Some details:

- For \( m \) granularities, the controller needs to create \( m \) different key pairs
- For \( n \) consumers, the controller needs to pack \( O(m) \) different keys for each of consumer

The number of granularities could increase exponentially as the name length increases
NAC with Attribute-Based Encryption

◊ Use attribute policy (String) as the encryption key
  o (Living-Room AND Display-Device) OR Display-Authorized

◊ Decrypt the content with sufficient attribute (key bits) set
  o Attribute set 1: Display-Authorized
  o Attribute set 2: Living-Room, Display-Device
NAC-ABE

(Living-Room AND Display-Device) OR Display-Authorized

Android Phone (data owner)

Attributes: Living-Room, Display-Device, Display-Authorized

Decryption Key (KDK)

Public Key

Content Key (C-KEY)

Data

Camera (data producer)

Television (data consumer)

Content Key (C-KEY)

Data
Access Manager (aka Data Owner)

- Encryption policy using public key (KEK)
  
  /home/controller/NAC
  /home/LivingRoom/Camera-01/FrontView/KEK/
  "(Living-Room AND Display-Device)
  OR
  Display-Authorized"

- Authorizes decryptors by publishing encrypted version of attribute set (KDK)
- Issued along with the identity Certificate

/home/controller/NAC-ATTR/ENCRYPTED-BY
/home/LivingRoom/Television/KEY/<key-id>
Encryptor (aka Producer)

From Access Manager / provisioned or dedicated data owner storage

- Fetches and stores KEK for the configured with access prefix

Interest ->
/home/controller/NAC
/home/LivingRoom/Camera-01/FrontView/KEK

- Generates (re-generates) symmetric Content Key (CK)
- Publishes CK data under configured namespace, encrypted by KEK

Data:
/home/LivingRoom/Camera-01/CK/<key-id>
/ENCRYPTED-BY
/home/controller/NAC/KEK/
“(Living-Room AND Display-Device) OR Display-Authorized”

- Encrypts input data using CK, returns encrypted content
- Exact name of the corresponding CK data is embedded in the encrypted content
Decryptor (aka Consumer)

- Fetch the encrypted Content Data
- Get the name of the corresponding CK: CK name is embedded in the encrypted content

Television
(data consumer)

From Encryptor / from same place as data

- Fetches CK data for the name extracted from input encrypted payload

Interest->
/home/LivingRoom/Camera-01/CK/<key-id>
Better Scalability

♦ In NAC-ABE, the complexity of key generation is $O(a)$, complexity of key distribution is $O(a*n)$, where
  o $a$ is the number of attributes (greatly smaller than the number of granularities)
  o $n$ is the number of consumers

♦ Some details
  o For $m$ granularities, the controller needs to create $m$ different attribute policies with $a$ attributes (with $a$ attributes, one can create more than $2^a$ policies)
  o For $n$ consumers, the controller needs to deliver $O(m)$ attribute keys to each consumer

♦ Improvement
  o Given $m$ is a fixed number for a system and a consumer’s attributes are decided by its identity, attributes can be issued with the issuance of NDN identity certificate: $O(m)$
NAC-ABE Main APIs

Access Controller

void
commandProducerPolicy(const Name& producerPrefix, const Name& dataPrefix, const std::string& policy);

Producer (Encryptor)

void
produce(const Name& dataName, const uint8_t* content, size_t contentLen);

Consumer (Decryptor)

void
consume(const Name& dataName, const Name& tokenIssuerPrefix);
Existing Integration Tests and Examples

◊ NAC Examples
  o [https://github.com/named-data/name-based-access-control/tree/new/examples](https://github.com/named-data/name-based-access-control/tree/new/examples)

◊ NAC-ABE Quick Start

◊ NAC-ABE Integration Tests