ACM ICN 2018 Tutorial
Boston, MA, September 21, 2018

Data Confidentiality and Access Control

Name-based Access Control

Zhiyi Zhang (UCLA)

Today's Confidentiality and Access Control

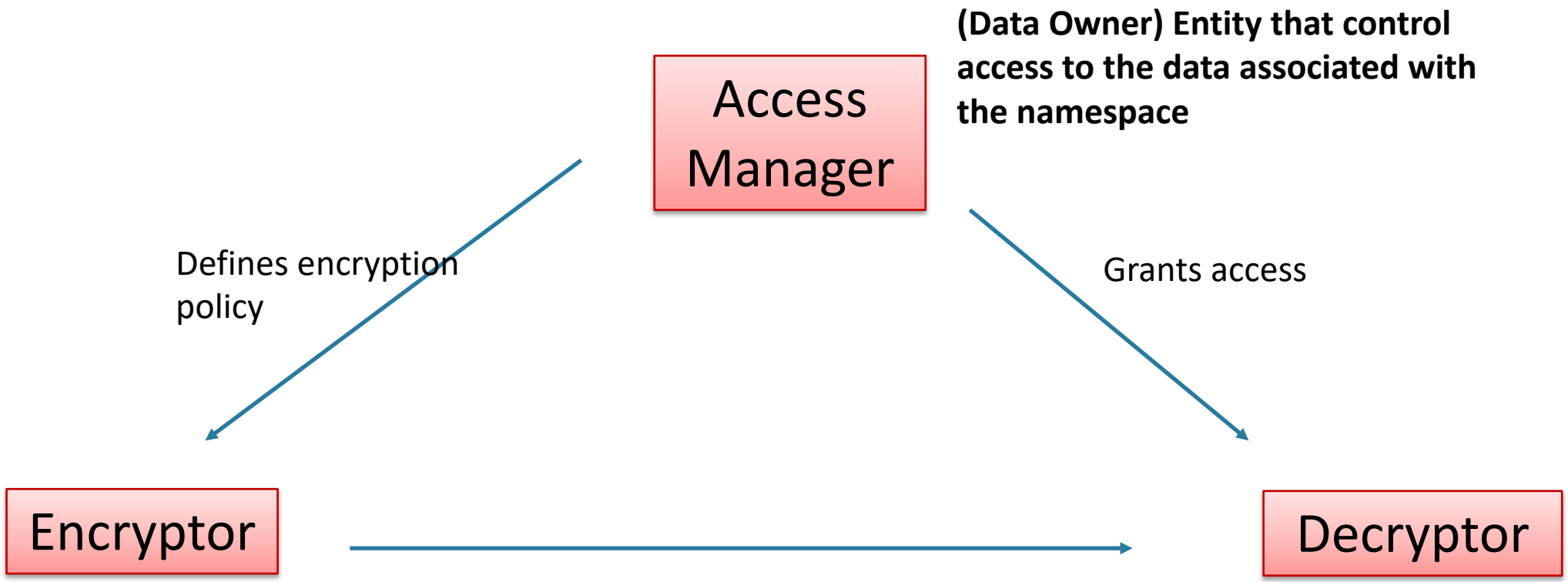
- Traditional connection-based confidentiality
  - End-to-End Confidentiality? (e.g., CDN, Middlebox)
  - Multi-party confidentiality and access control?

- Encrypt Content Directly over TCP/IP architecture
  - Inefficient Key Distribution (where to fetch those keys?)
  - Content Multicast

Confidentiality and Access Control over NDN

- Built-in Data Authentication and Integrity
  - Integrity helps the confidentiality (e.g., MITM)
- Automating key distribution with naming conventions
  - Extract decryption key name from content and Data name
  - Fetch decryption key directly using name
- Content Delivery and In-network Cache

Name-Based Access Control (NAC) Concepts



Name-Based Access Control (NAC) Concepts

Access Controller– Android Phone

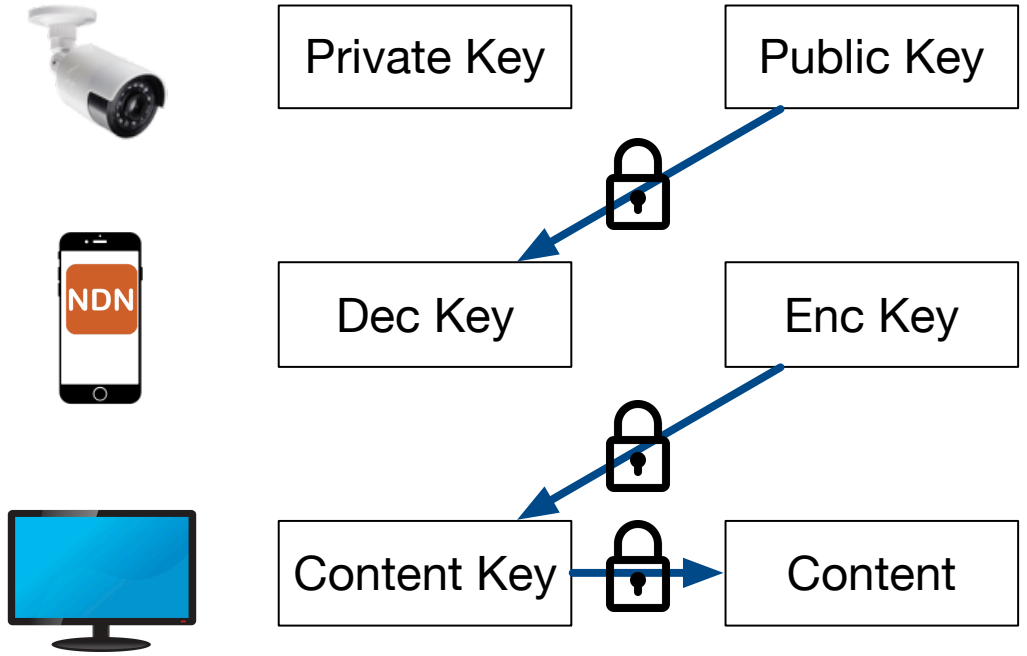
- ◊ Creates a list of encryption/decryption key pairs
- ◊ Controls which encryption keys are used to encrypt which namespace
- ◊ Control whom to distribute the corresponding decryption keys

Producers (Encryptors)– Camera

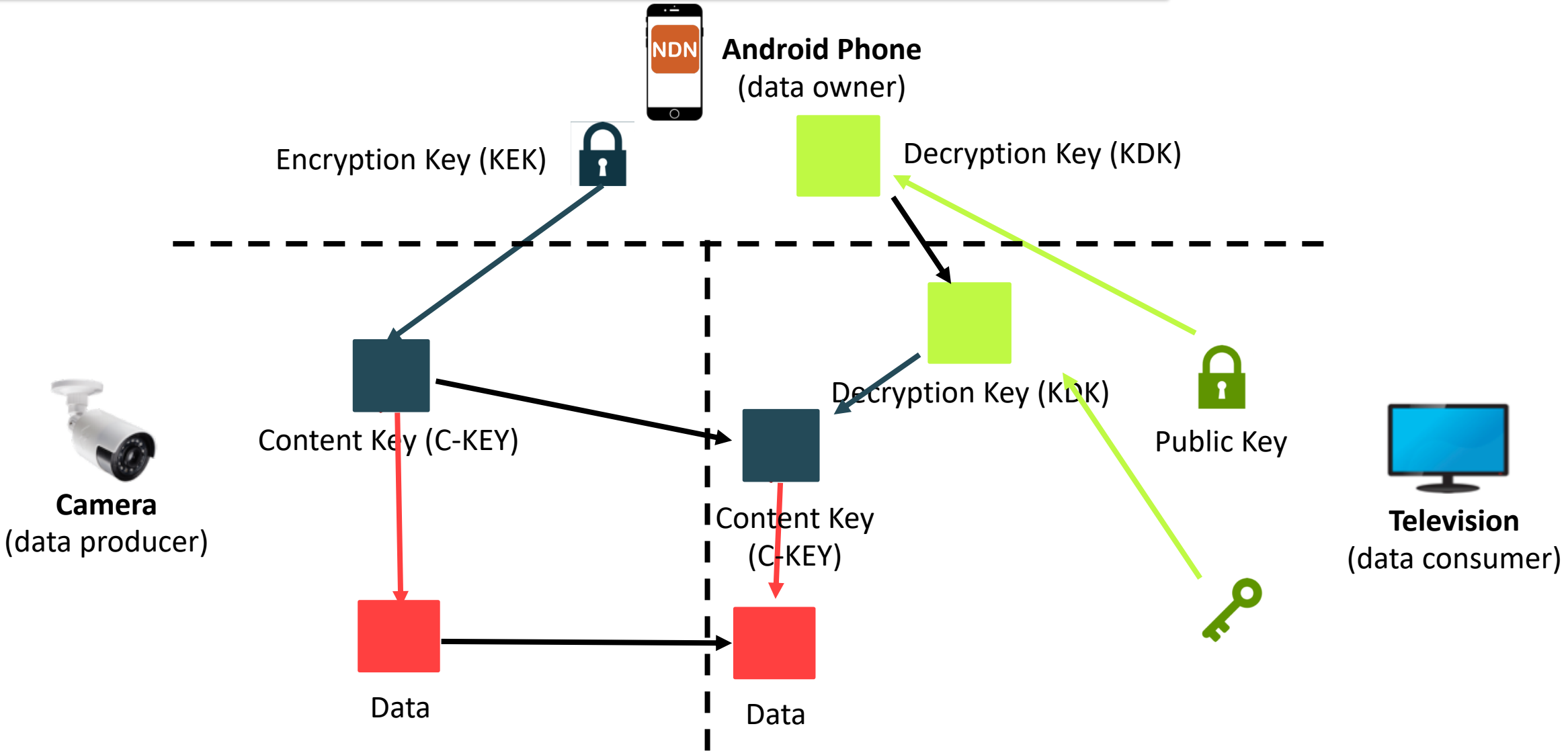
- ◊ Fetch the right encryption keys to encrypt data (2 steps)

Consumers (Decryptor) –Television

- ◊ Fetch the right decryption keys to decrypt data (3 steps)



NAC Process


NDN
Android Phone
(data owner)
Encryption Key (KEK)
Decryption Key (KDK)
Content Key (C-KEY)
Decryption Key (KDK)
Public Key
Camera
(data producer)
Content Key
(C-KEY)
Television
(data consumer)
Data
Data

An example of NAC's naming convention

Control the access to the smart home camera data

- The owner/controller of the access control system
  - /home/controller
- The authorized consumer:
  - /home/LivingRoom/Television-01
- The dataset that is being controlled:
  - /home/LivingRoom/Camera-01/FrontView
- The producer:
  - /home/LivingRoom/Camera-01

Access Manager (aka Data Owner)

Android Phone
(Access Manager)

- 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)

Camera
(data 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

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

- 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)

Television
(data consumer)

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

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>

- 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 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 encrypto(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?

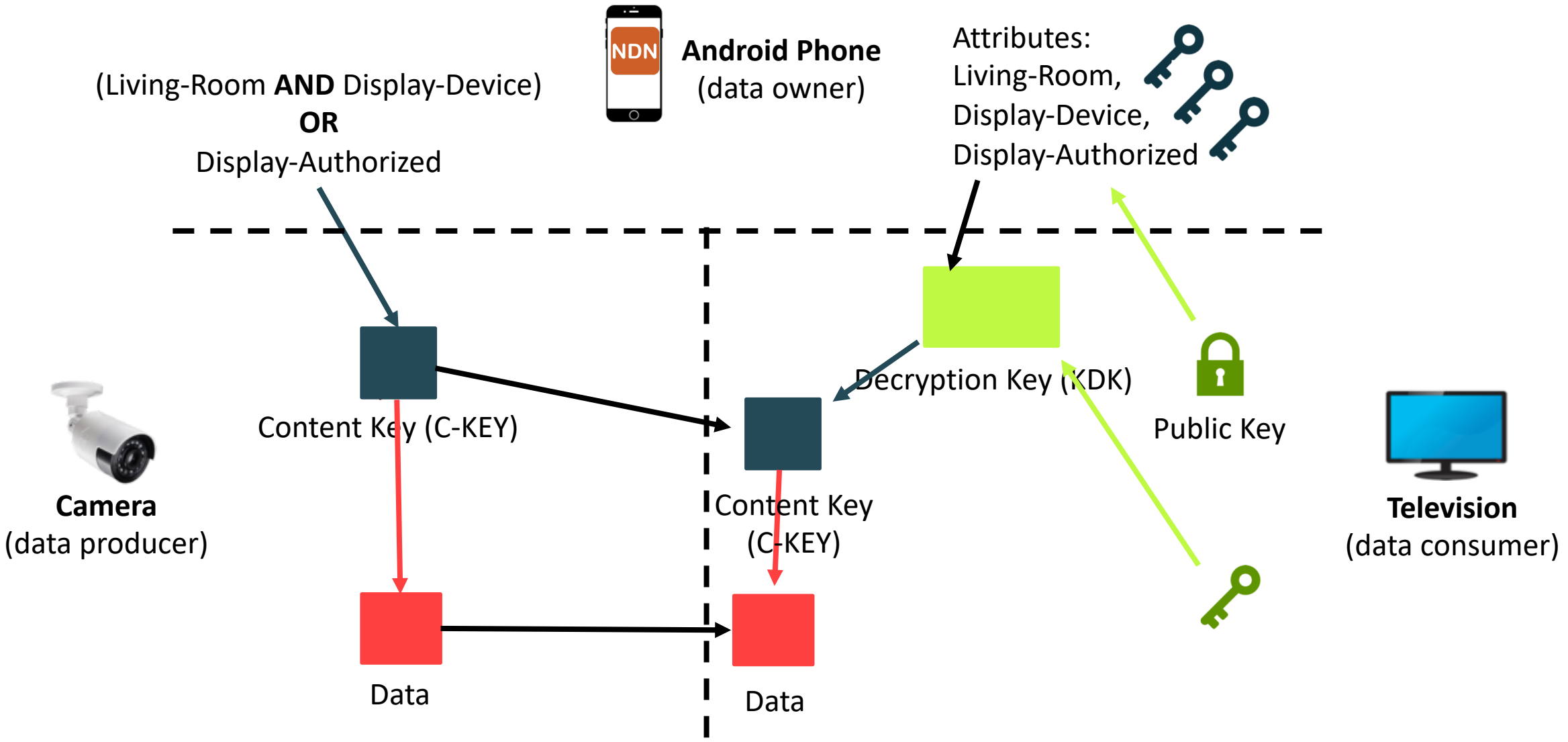
- In NAC, the complexity of key generation is O(m), and complexity of key distribution is O(m*n) where
  - m is the number of granularities
  - n is the number of consumers
- Some details:
  - For m granularities, the controller needs to create m different key pairs
  - For n consumers, the controller 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
  - (**Living-Room** AND **Display-Device**) OR **Display-Authorized**
- Decrypt the content with sufficient attribute (key bits) set
  - Attribute set 1: Display-Authorized
  - Attribute set 2: Living-Room, Display-Device

NAC-ABE


(Living-Room **AND** Display-Device)
OR
Display-Authorized
NDN
Android Phone
(data owner)
Attributes:
Living-Room,
Display-Device,
Display-Authorized
Content Key (C-KEY)
Decryption Key (KDK)
Public Key
Camera
(data producer)
Content Key
(C-KEY)
Television
(data consumer)
Data
Data

Access Manager (aka Data Owner)

Android Phone
(Access Manager & Attribute Authority)

- 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 Certifcate

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

Encryptor (aka Producer)

From Access Manager / provisioned or dedicated data owner storage

Camera
(data producer)

- Fetches and stores KEK for the configured with access prefix

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

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

- 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"

Decryptor (aka Consumer)

Television
(data consumer)

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

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
  - a is the number of attributes (greatly smaller than the number of granularities)
  - n is the number of consumers
- Some details
  - 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)
  - For n consumers, the controller needs to delivery O(m) attribute keys to each consumer
- Improvement
  - 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
  - https://github.com/named-data/name-based-access-control/tree/new/examples
- NAC-ABE Quick Start
  - https://github.com/Zhiyi-Zhang/NAC-ABE
- NAC-ABE Integration Tests
  - https://github.com/Zhiyi-Zhang/NAC-ABE/tree/master/tests/integrated-tests