Authorization credentials for controlled sharing in NDN: Experiments with codecaps and macaroons in NDN.JS

NDNCOMM 2014

Pedro de-las-Heras-Quirós, Eva M. Castro-Barbero
<pedro.delasheras@urjc.es>

Information Technology and Communications Department
Universidad Rey Juan Carlos, Spain

September 5, 2014

Contents

Introduction

- We are developing prototypes of codecaps and macaroons for NDN using NDN-CCL (NDN.JS v0.3), ndnd-tlv, ndncert, Mini-CCNx (adapted to ndnd-tlv)
- Work in progress to explore potential solutions for encryption based group access control for NDN apps
- Expect more doubts than claims:
  - Can these mechanisms improve consumer anonymity in NDN when compared with signed interests?
  - Can they facilitate service composition of NDN apps? Example applications:
    - Raw sensor data stored and published to service that transforms and republishes data to different group, with different rights
    - Example application: want to let my family group / friends group some of the photos in NDNFlickr, withouth them having an account there
    - Delegation of voting rights acording to subject
    - Open mHealth

Contenidos

What are Codecaps

Secure abstraction with code capabilities. R. van Renesse, H.D. Johansenn, N. Naigaonkar and D. Johansen. In 21st Euromicro International Conference on Parallel, Distributed and Network-Based Processing, 2013

- Codecaps are Capabilities that embed code that programatically expresses the rights acquired by the owner
- Rights are code (Javascript in the original paper) that is evaluated in the context of a request to grant/deny access.

What are Codecaps

Codecaps can be extended by principals.

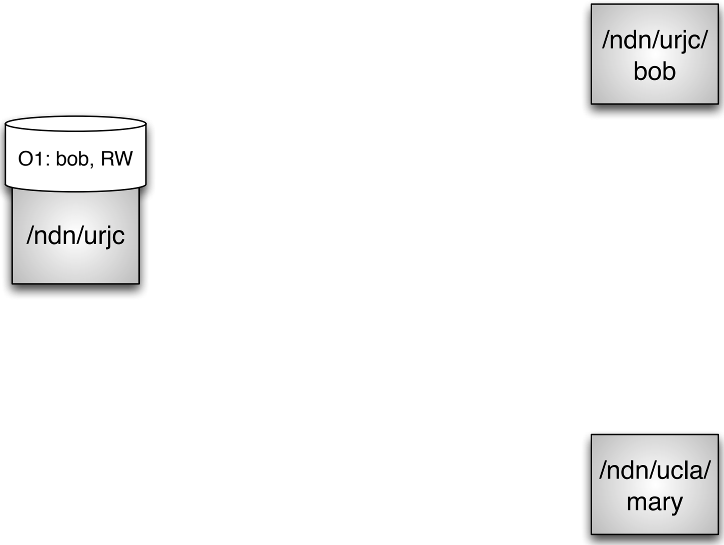
- Each codecap includes a certificate chain that can be extended by its owner by adding new right functions that attenuate the original rights
- Codecaps are extended for a particular principal: each certificate in the chain signs both a new right function and the Public Key of the principal who can use the new extended codecap

What are Codecaps

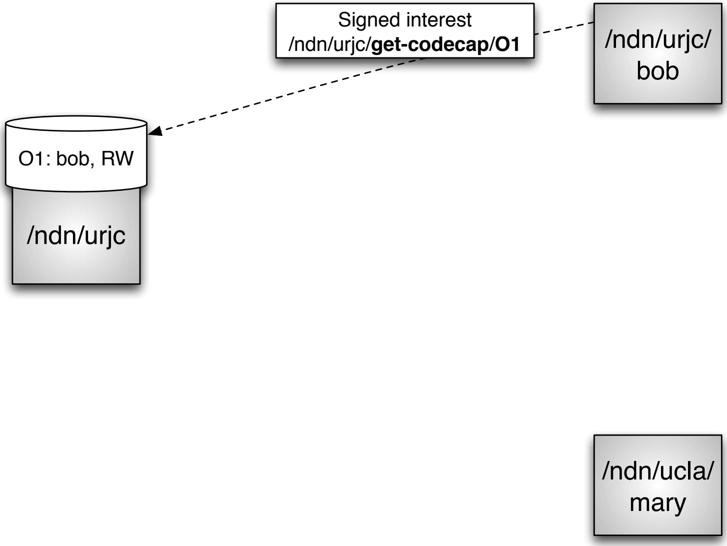
A request includes a codecap + action requested

- A request can be created by any principal owning a Codecap by signing the requested action with its private key and sending it alongside the codecap
- The original creator of the codecap validates the chain of certificates of the request, and evaluates if every rights function is satisfied in the context of the request, granting or denying access

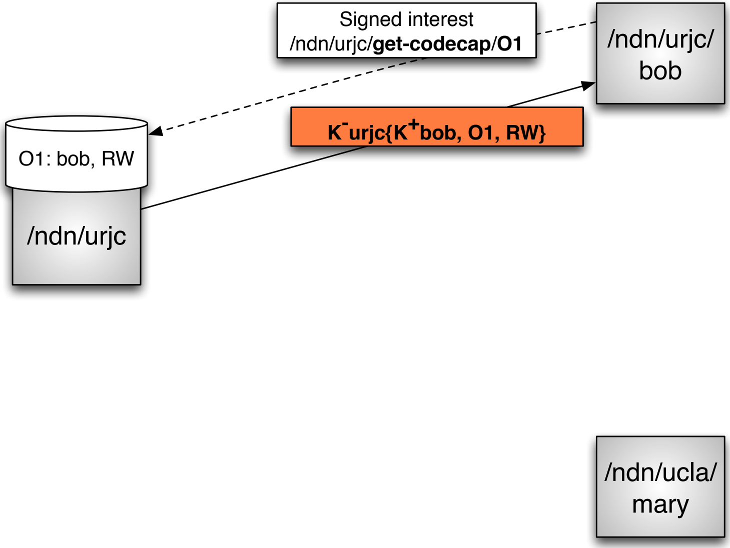
Example



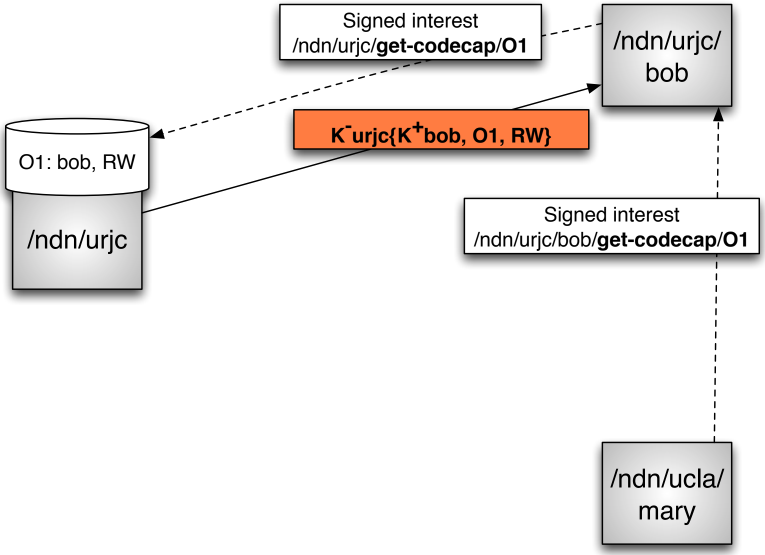
Example


Signed interest
/ndn/urjc/**get-codecap**/**O1**
/ndn/urjc/bob
O1: bob, RW
/ndn/urjc
/ndn/ucla/mary

Example

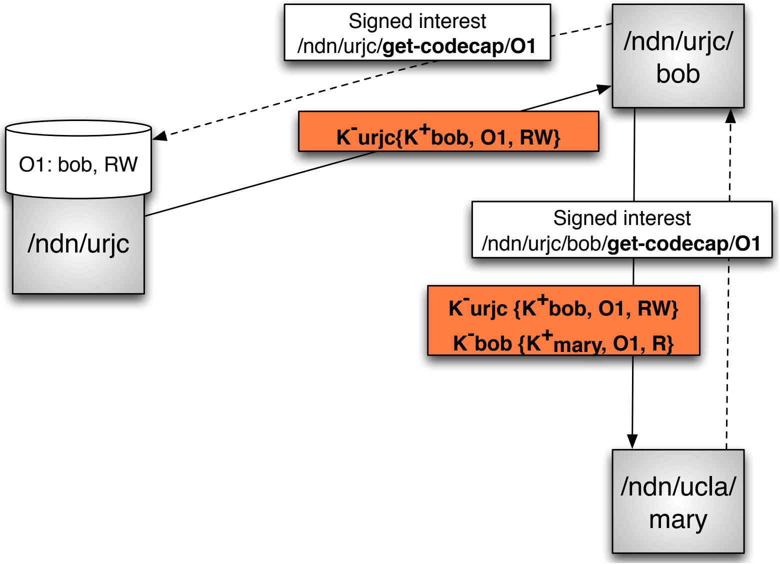

Signed interest
/ndn/urjc/**get-codecap/O1**
/ndn/urjc/
bob
O1: bob, RW
/ndn/urjc
K-urjc{K+bob, O1, RW}
/ndn/ucla/
mary

Example

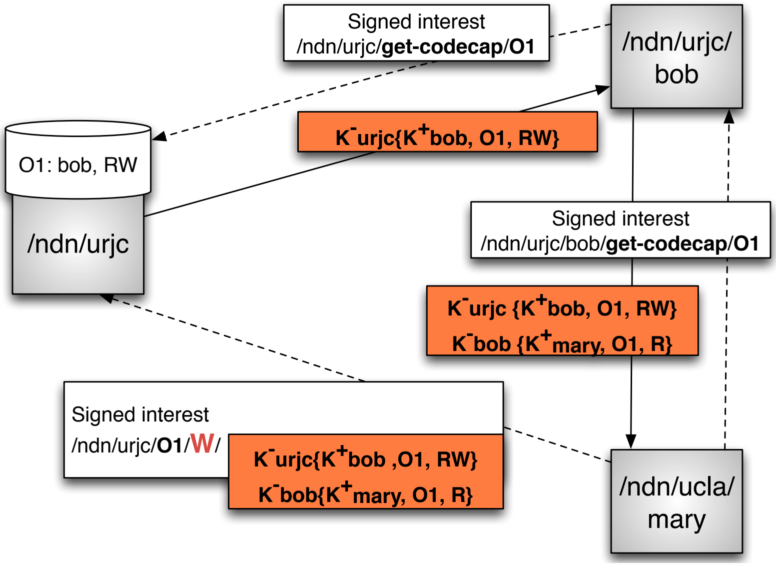


Signed interest
/ndn/urjc/**get-codecap/O1**
/ndn/urjc/
bob
O1: bob, RW
/ndn/urjc
K-urjc{K+bob, O1, RW}
Signed interest
/ndn/urjc/bob/**get-codecap/O1**
/ndn/ucla/
mary

Example


Signed interest
/ndn/urjc/get-codecap/O1
/ndn/urjc/bob
K⁻urjc{K⁺bob, O1, RW}
O1: bob, RW
/ndn/urjc
Signed interest
/ndn/urjc/bob/get-codecap/O1
K⁻urjc {K⁺bob, O1, RW}
K⁻bob {K⁺mary, O1, R}
/ndn/ucla/mary

Example


Signed interest
/ndn/urjc/**get-codecap/O1**
/ndn/urjc/
bob
K-urjc{K+bob, O1, RW}
O1: bob, RW
/ndn/urjc
Signed interest
/ndn/urjc/bob/**get-codecap/O1**
K-urjc {K+bob, O1, RW}
K-bob {K+mary, O1, R}
Signed interest
/ndn/urjc/**O1**/W/
K-urjc{K+bob ,O1, RW}
K-bob{K+mary, O1, R}
/ndn/ucla/
mary

Contenidos

What are Macaroons

Macaroons: Cookies with contextual caveats for decentralized authorization in the cloud. A. Birgisson, J.G. Politz, Úlfar Erlingsson, A. Taly, M. Vrable, and M. Lentczner. In Network and Distributed System Security Symposium, 2014

- Similar to codecaps although they're not capabilities, but credentials

What are Macaroons

Also embed code: authorization predicates in caveats, similar to rights functions of codecaps

- Express when, where, by who and for what purpose a producer principal should authorize requests for content or services it owns

What are Macaroons

Macaroons can also be extended but they don't use PK certificates for expressing delegation

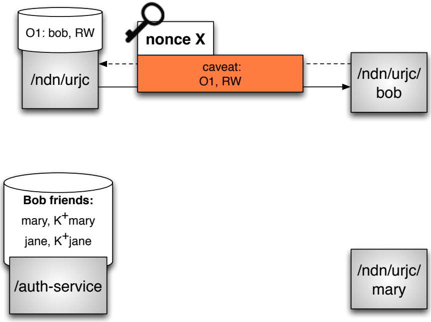
- The list of caveats added by principals is chained through HMAC's: much more efficient, and potentially anonymous for NDN consumers
- Original creator of macaroon keeps secret the root key used to calculate the first HMAC, and adds nonce identifying it to macaroon
- Next principal in chain will use the previous HMAC as the key for calculating HMAC of next caveat added
- Requests can only be validated by original creator, who recalculates the chain of HMACs starting with secret root key indexed by nonce in macaroon of request

What are Macaroons

Main innovation of macaroons: third-party caveats

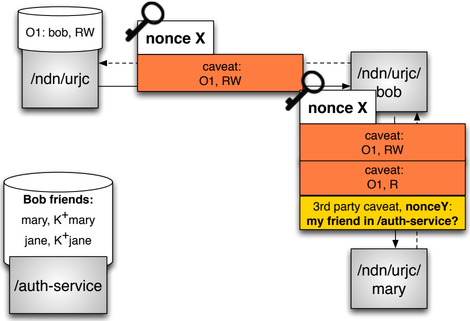
Example

Bob receives macaroon created by /ndn/urjc



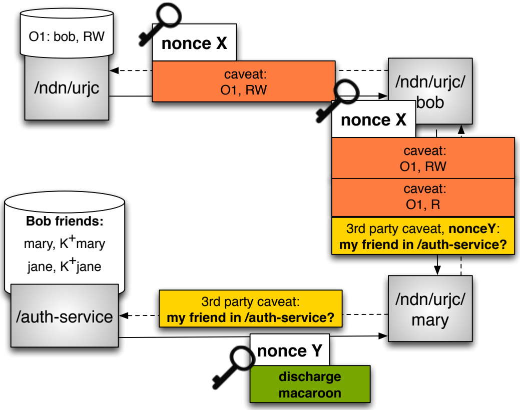
Example

Bob extends the macaroon with normal caveat and with third-party caveat that requires Mary to authenticate in auth-service, and then sends the extended macaroon to Mary



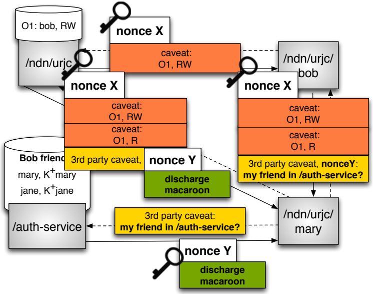
Example

In order for Mary to create a request for O1, she must first authenticate herself in the third party auth-service to satisfy the third party caveat as demanded by Bob



Example

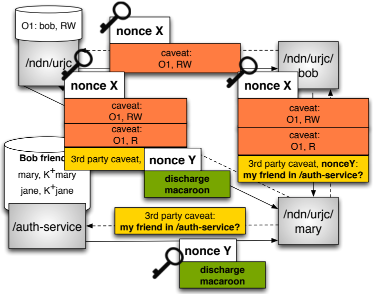
- Mary then adds the discharge macaroon to a request sent to /ndn/urjc



Example

- /ndn/urjc can validate first party caveats in the context of the request, and can validate the third party caveat imposed by Bob checking the presence of the discharge macaroon

Without /ndn/urjc knowing neither what the requirement was, who was the third party, or who the consumer is!



Current work: Adding keys to macaroon

- Encryption based group access control: producer adds session keys to macaroon for encrypting interests received and data sent
  - Data packet sent encrypted uses intrinsic multicast of data packets
- By adding the public key of the producer to the macaroon we enable different trust models where those receiving a macaroon from an intermediary principal can verify content of the original producer by trusting the macaroon

Current work: Revocation of macaroons

- By frequently revoking macaroons, which are cheap to create:
  - Producer can frequently change the session keys
  - Producer can frequently change its PK, increasing anonimity of data producers
- How-to revocation of macaroons / session keys / public keys:
  - Directories of macaroons (scaling through hierarchy) + versioning of data + frequent expiration (it is inexpensive to generate new macaroons)

Contenidos

Authorization credentials vs. ACLs

- More scalability: producer does not store an amount of state proportional to the number of consumer principals as must be done with ACLs
- More flexibility: each intermediate principal can design its access control policy before delegating, not constrained by fixed set of policies predefined by the original producer

Comparison Codecaps / Macaroons

- Codecaps and third-party caveats of macaroons enable flexible service composition in NDN apps
- Both support restricted delegation, confinement and revocation
- When using codecaps anonimity of consumers is lost: producers must know and trust consumers' PK to validate a codecap
- By using HMAC's + symmetric-key encryption, macaroons enable consumer anonymity
- By using HMAC's instead of PK certificates, macaroons are more efficient both at creation time and at validation time, enabling frequent revocation

Future Work

- Increase information obscurity through combinations of encrypted namespaces + multi-key searchable encryption of both, encrypted directories and encrypted producer data stores
- Control access for anycast: producer replicas sharing the root key of macaroons
- Feedback from NDN + crypto experts about the validity of our adaptations made to original codecaps/macaroons
- It is our first NDN library. Need to improve current code based on NDN.JS v0.3: improve codification with Protocol Buffers, adaptation to newest version of NDN.JS (keychain), ...
- Porting to: Firefox plugin, NDN-CCL C++/Python and to NDN-CXX