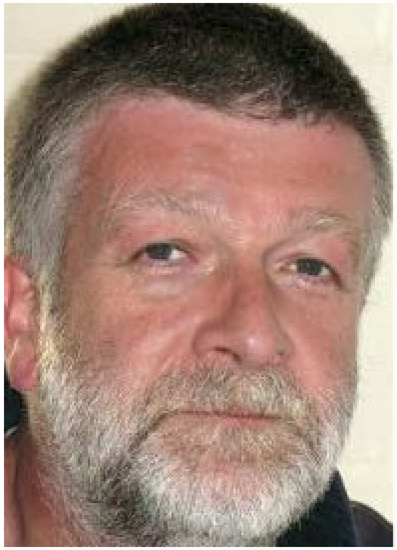
Mobile and Secure Healthcare: Encrypted Objects and Access Control Delegation

January 28, 2016

SNIA™
Cloud Storage Initiative

SNIA Presenters



Alex McDonald
Chair - SNIA
Cloud Storage
NetApp

Martin Rosner
Standardization Officer
Phillips

David Slik
Co-Chair SNIA Cloud
Technical Work Group
Technical Director for Object
Storage, NetApp

SNIA Legal Notice

SNIA™
Cloud Storage Initiative

- The material contained in this tutorial is copyrighted by the SNIA unless otherwise noted.
- Member companies and individual members may use this material in presentations and literature under the following conditions:
  - Any slide or slides used must be reproduced in their entirety without modification
  - The SNIA must be acknowledged as the source of any material used in the body of any document containing material from these presentations.
- This presentation is a project of the SNIA Education Committee.
- Neither the author nor the presenter is an attorney and nothing in this presentation is intended to be, or should be construed as legal advice or an opinion of counsel. If you need legal advice or a legal opinion please contact your attorney.
- The information presented herein represents the author's personal opinion and current understanding of the relevant issues involved. The author, the presenter, and the SNIA do not assume any responsibility or liability for damages arising out of any reliance on or use of this information.

  NO WARRANTIES, EXPRESS OR IMPLIED. USE AT YOUR OWN RISK.

Outline

1. Secure object mobility and access control
2. Challenges in the healthcare industry
3. Example Implementation using new extensions
4. Scenarios

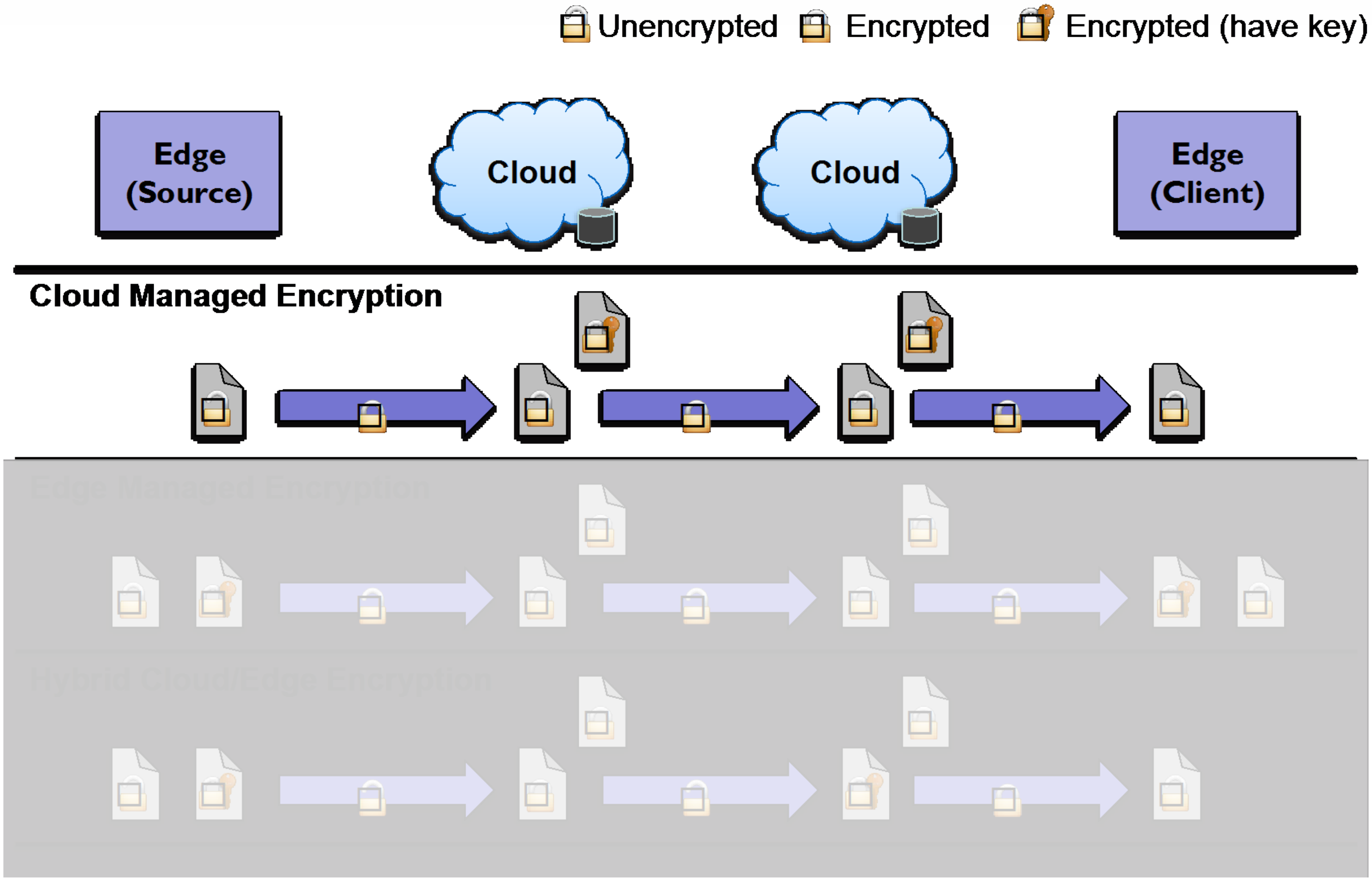
Data Security in the Cloud

SNIA™
Cloud Storage Initiative

- Organizations (and individuals) are increasingly concerned about storing unencrypted data in the cloud
  - What happens if the cloud provider is compromised?
  - What happens if the cloud account is compromised?
  - What happens if a system that can access the cloud account is compromised?
  - What happens if the cloud provider goes out of business?
- All of these scenarios can result in massive data breaches, which are often undetected due to lack of audit

Visual Taxonomy

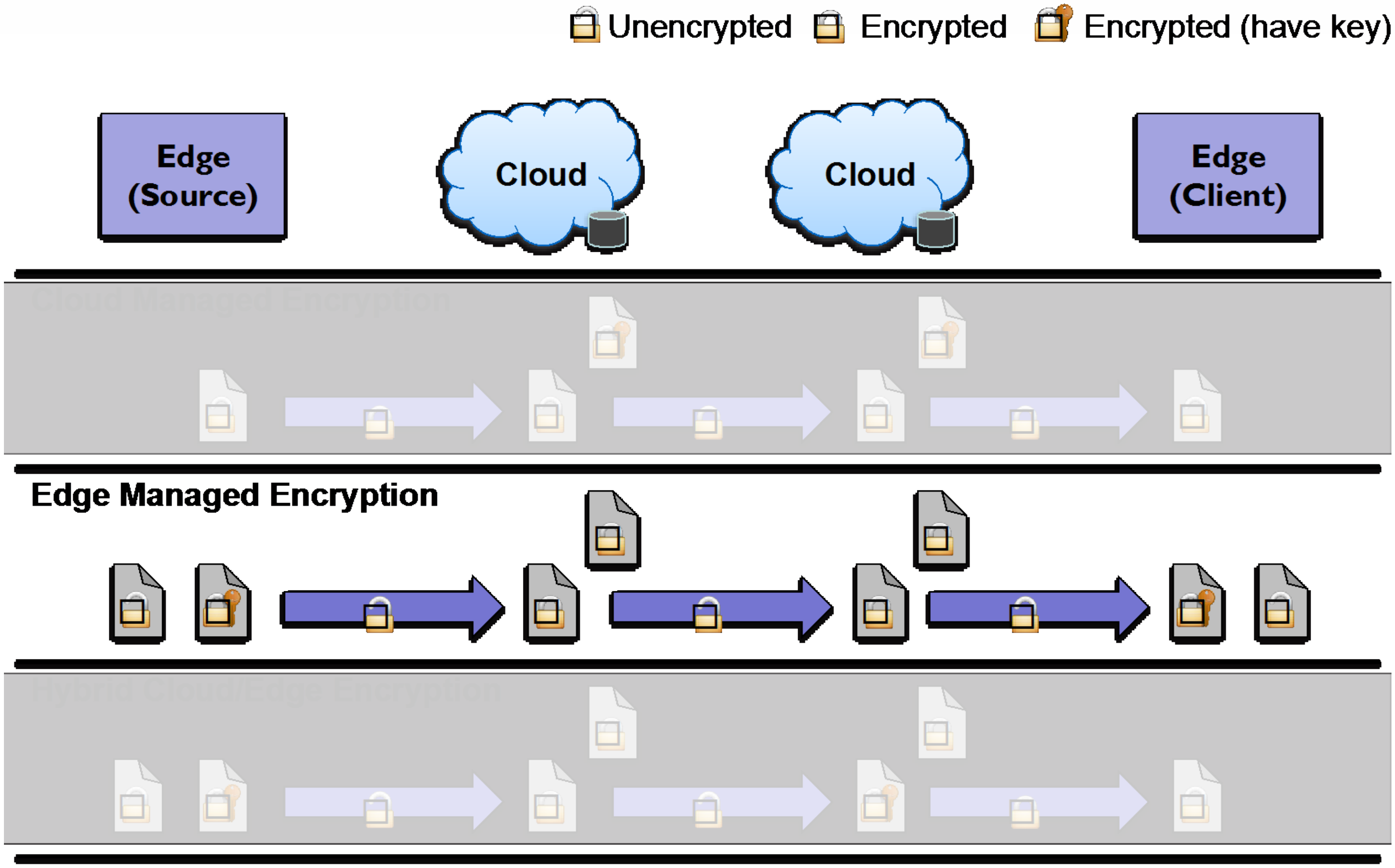
SNIA™
Cloud Storage Initiative



Cloud-Managed Encryption

- Advantages:
  - Simplest approach
  - Unmodified clients, don’t have to know about keys
- Disadvantages:
  - Cloud provider knows the keys
  - Cloud compromise allows bypass of access controls
  - Cloud compromise allows bypass of audit
  - Inefficient multiple encryption/decryption operations for both in-flight and at-rest

Visual Taxonomy



Unencrypted Encrypted Encrypted (have key)

Edge (Source)

Cloud

Cloud

Edge (Client)

Cloud Managed Encryption

Edge Managed Encryption

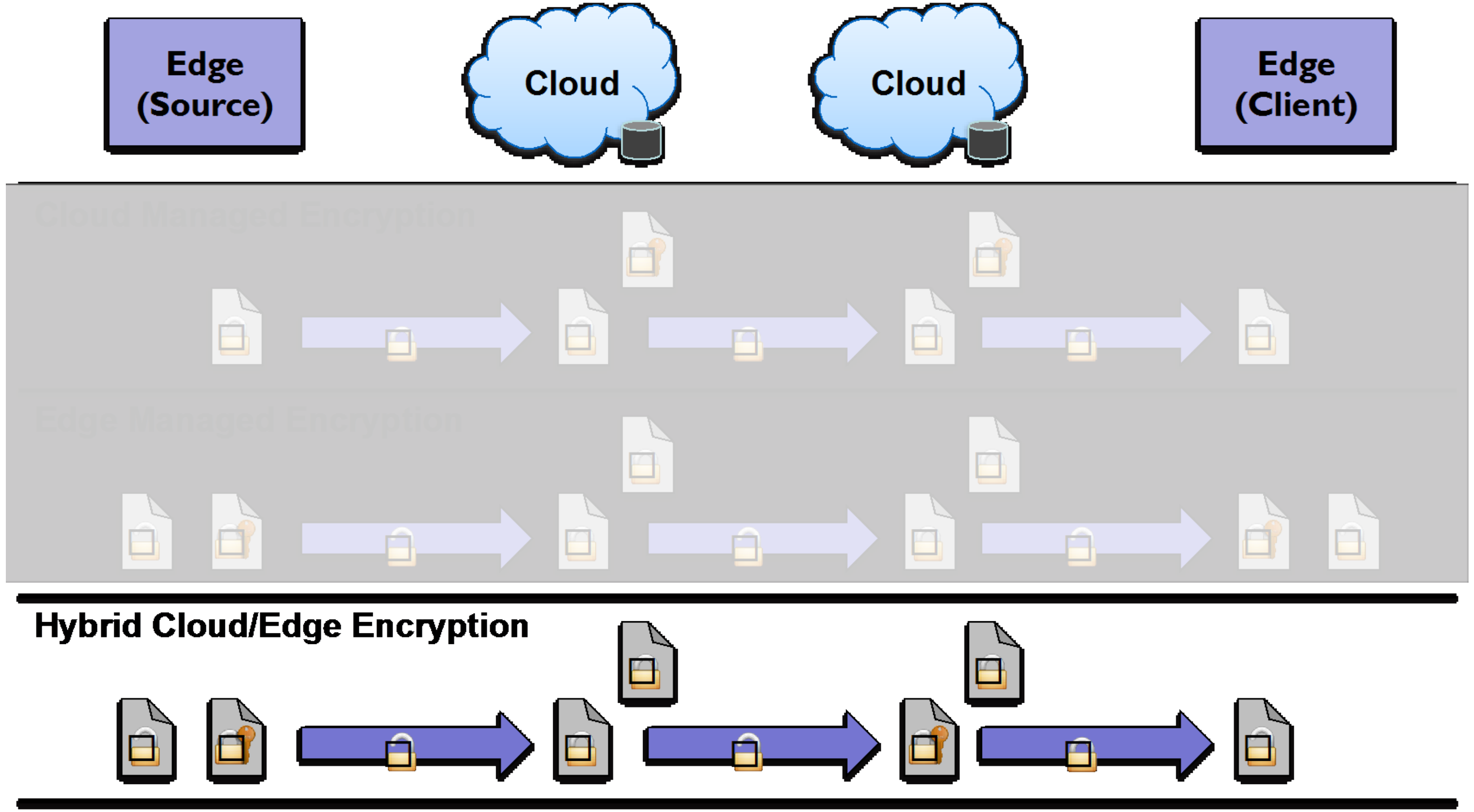
Hybrid Cloud/Edge Encryption

Edge-Managed Encryption

- Advantages:
  - Keys remain private, cloud provider never sees keys
  - Cloud compromise does not compromise security.
  - Can assume all cloud data is public
  - Most efficient, only encrypt once
  - Audit for all accesses
- Disadvantages:
  - Most complex approach
  - All edge systems accessing the data must be aware of and participate in key management.

Visual Taxonomy

🔓 Unencrypted 🔒 Encrypted 🔑 Encrypted (have key)

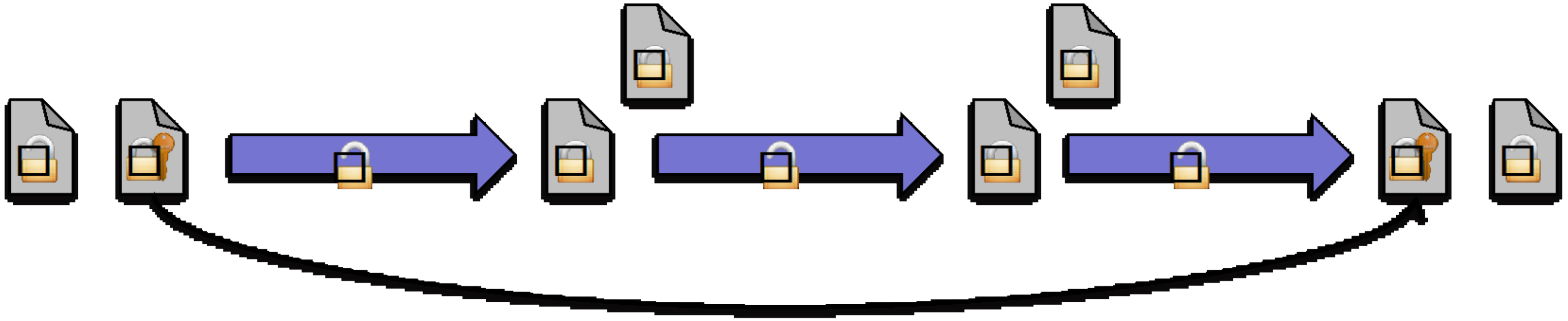


Hybrid Cloud/Edge Encryption

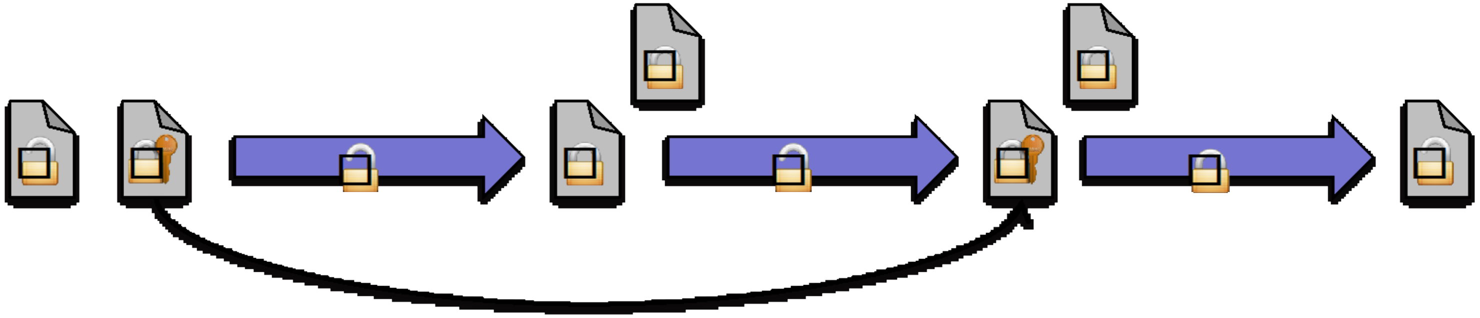
- Advantages:
  - Edge (cloud client) owns and manages the keys
  - Cloud can get access to keys if/when needed
  - Edge can make access control decisions
  - Clients can access plaintext or ciphertext
  - Edge can revoke access
  - Edge can audit all accesses
- Disadvantages:
  - Requires that the cloud software be trusted
  - Needs protocol for cloud/edge key exchange

Current Gaps in Standards

- How to securely send keys from the Source to the Client



- How to securely send keys from the Source to the Cloud



What do we need to standardize?

- Common themes
  - Need an “over the wire” self-describing encrypted object format
  - Need a way to distinguish plaintext vs. ciphertext requests
  - Need a way to securely request a key
  - Need a way to securely receive a key
  - Need a way to securely specify auditable events
  - Need a way to securely report back audit events

How is this being standardized?

- Encrypted Object CDMI Extension
  - Defines capabilities for server-side encryption and decryption
  - Defines standardized format for encrypted objects
    - Mime-type (or transfer encoding)
    - CMS or JOSE under consideration
    - Encryption of metadata

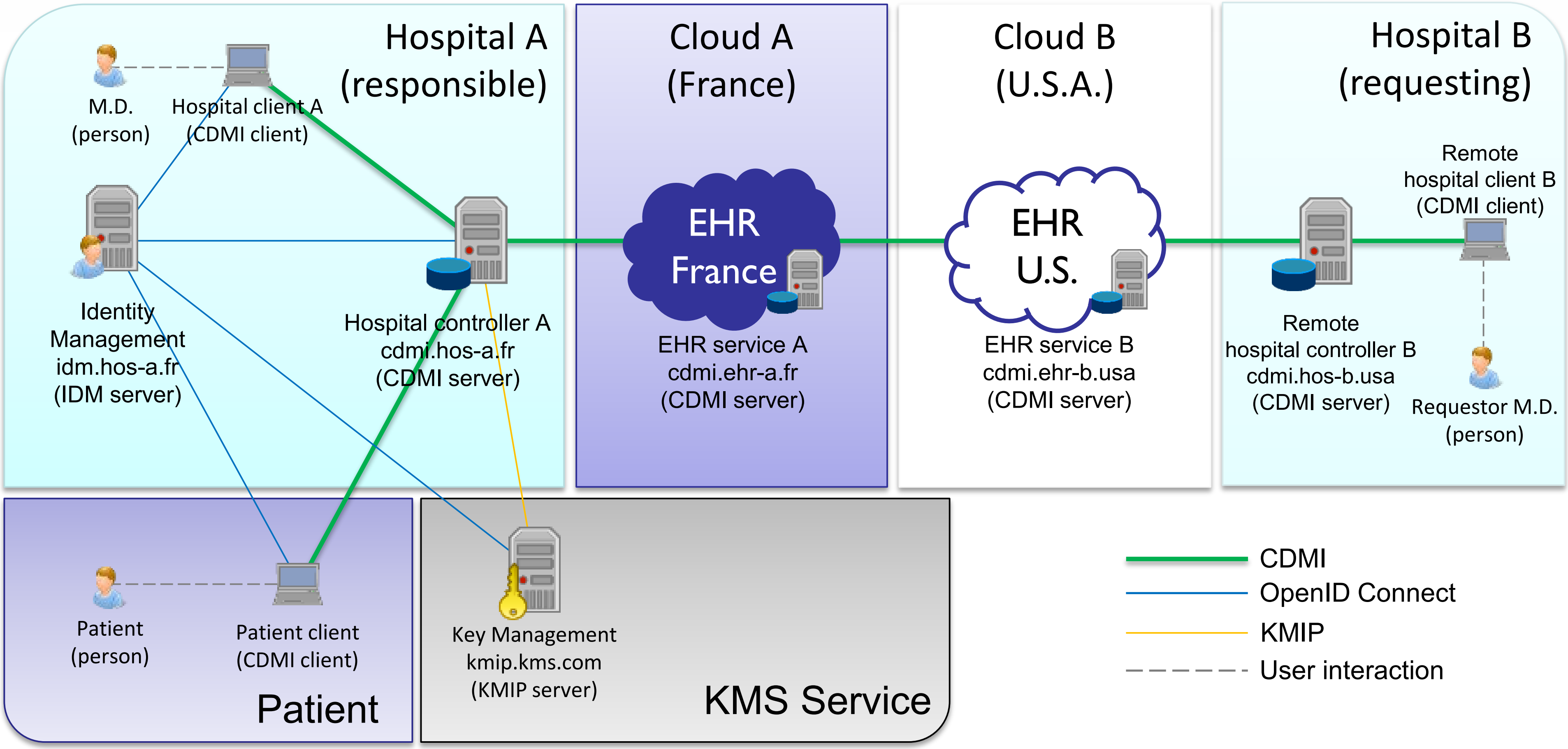
How is this being standardized?

- Delegated Access Control CDMI Extension
  - Defines capabilities for Delegated Access Control
  - Metadata to specify Delegated Access Control
    - Originator URL & Certificate
    - Metadata for Key Lookup
  - Plaintext and key access methods
    - Credentials, request message, response message
  - Optional redirection to key-per-request object
  - Key expiry and caching controls
  - Audit requirements

Challenges in the healthcare industry

- Slow moving
- Full of legacy
- Varying regulatory requirements
- Unclear ownership and governance of data
- Strict privacy requirements or expectations
- Changing patient consent expectations
- Government mandates for use of standards

Example Healthcare CDMI-based Implementation



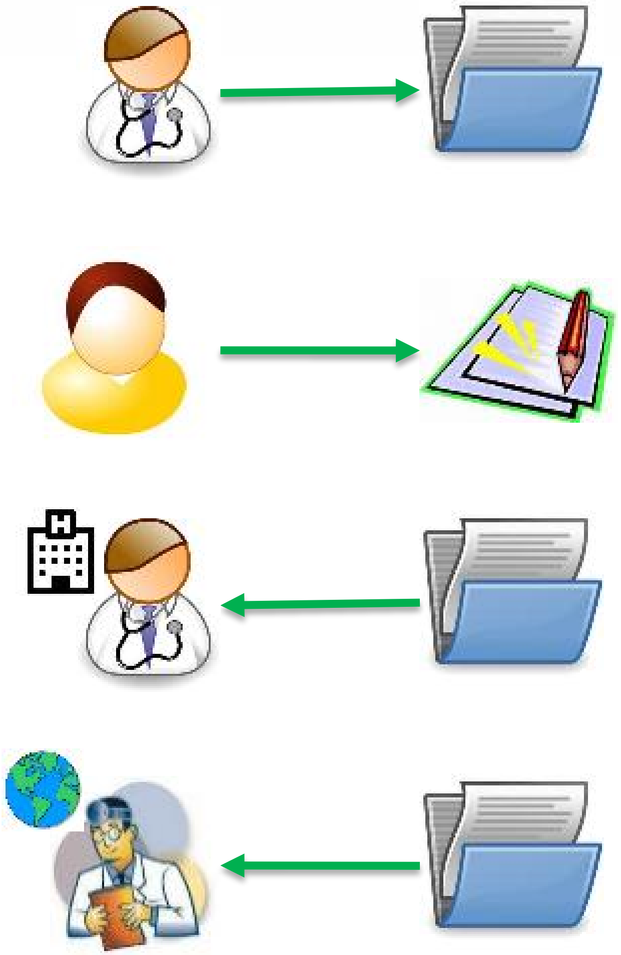
Hospital A (responsible)
M.D. (person)
Hospital client A (CDMI client)
Identity Management idm.hos-a.fr (IDM server)
Hospital controller A cdmi.hos-a.fr (CDMI server)
Cloud A (France)
EHR France
EHR service A cdmi.ehr-a.fr (CDMI server)
Cloud B (U.S.A.)
EHR U.S.
EHR service B cdmi.ehr-b.usa (CDMI server)
Hospital B (requesting)
Remote hospital client B (CDMI client)
Remote hospital controller B cdmi.hos-b.usa (CDMI server)
Requestor M.D. (person)
Patient (person)
Patient client (CDMI client)
Patient
Key Management kmip.kms.com (KMIP server)
KMS Service
CDMI
OpenID Connect
KMIP
User interaction

Scenario assumptions

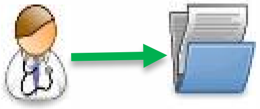
- Version control of health records is synchronized across clouds
- Construction of medical records from one or more CDMI-objects
- Determination of CDMI object-id required for particular patient
- Contractually established trust networks
- Transport layer security

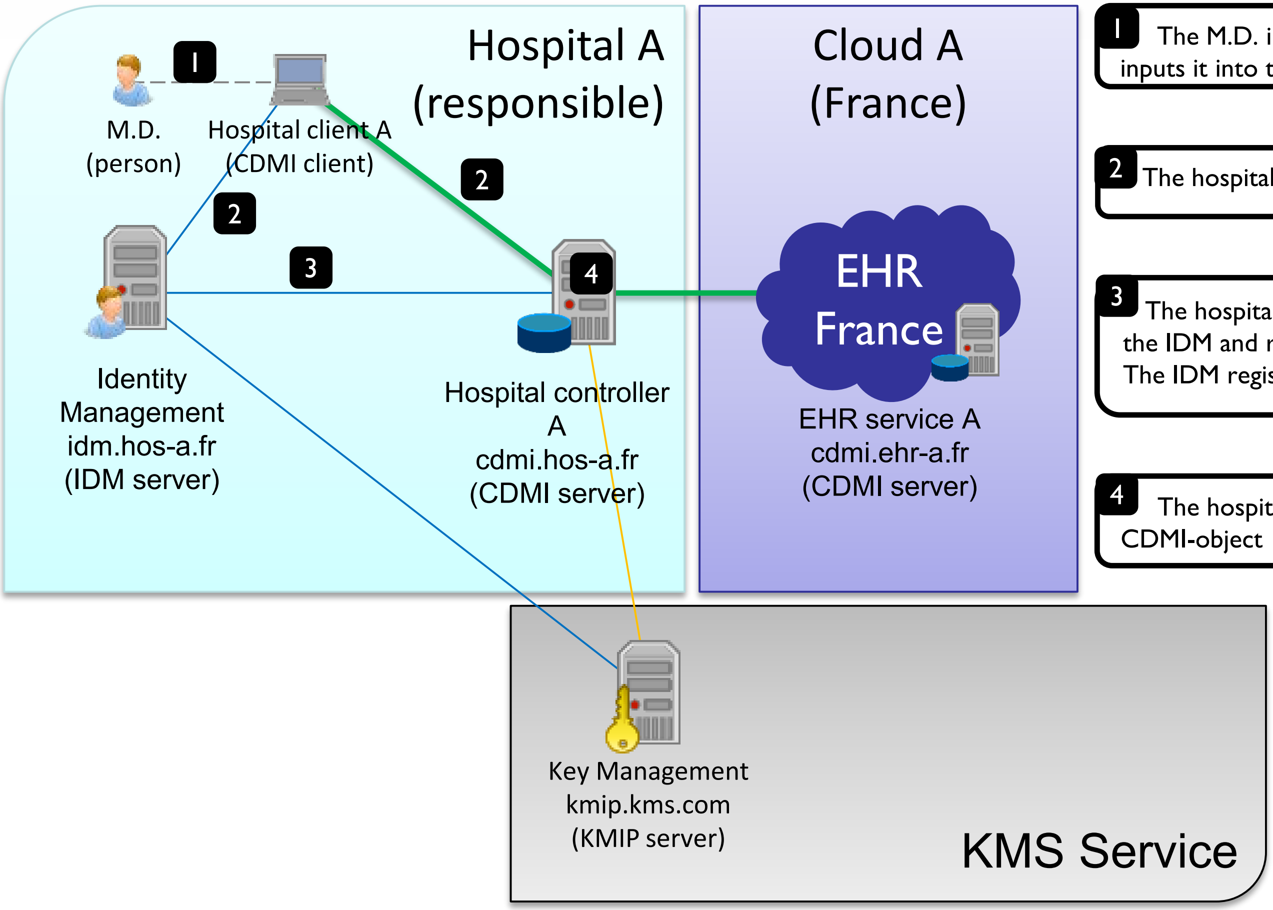
Medical Record Sharing Scenarios

1. Doctor creates a medical record
2. Patient creates consent directive
3. Local doctor requests medical records
4. Remote doctor requests medical records



Scenario 1: Creation of Medical Record (1 of 2)




Hospital A (responsible)
Cloud A (France)
M.D. (person)
Hospital client A (CDMI client)
1
2
2
3
4
Identity Management idm.hos-a.fr (IDM server)
Hospital controller A cdmi.hos-a.fr (CDMI server)
EHR France
EHR service A cdmi.ehr-a.fr (CDMI server)
Key Management kmip.kms.com (KMIP server)
KMS Service

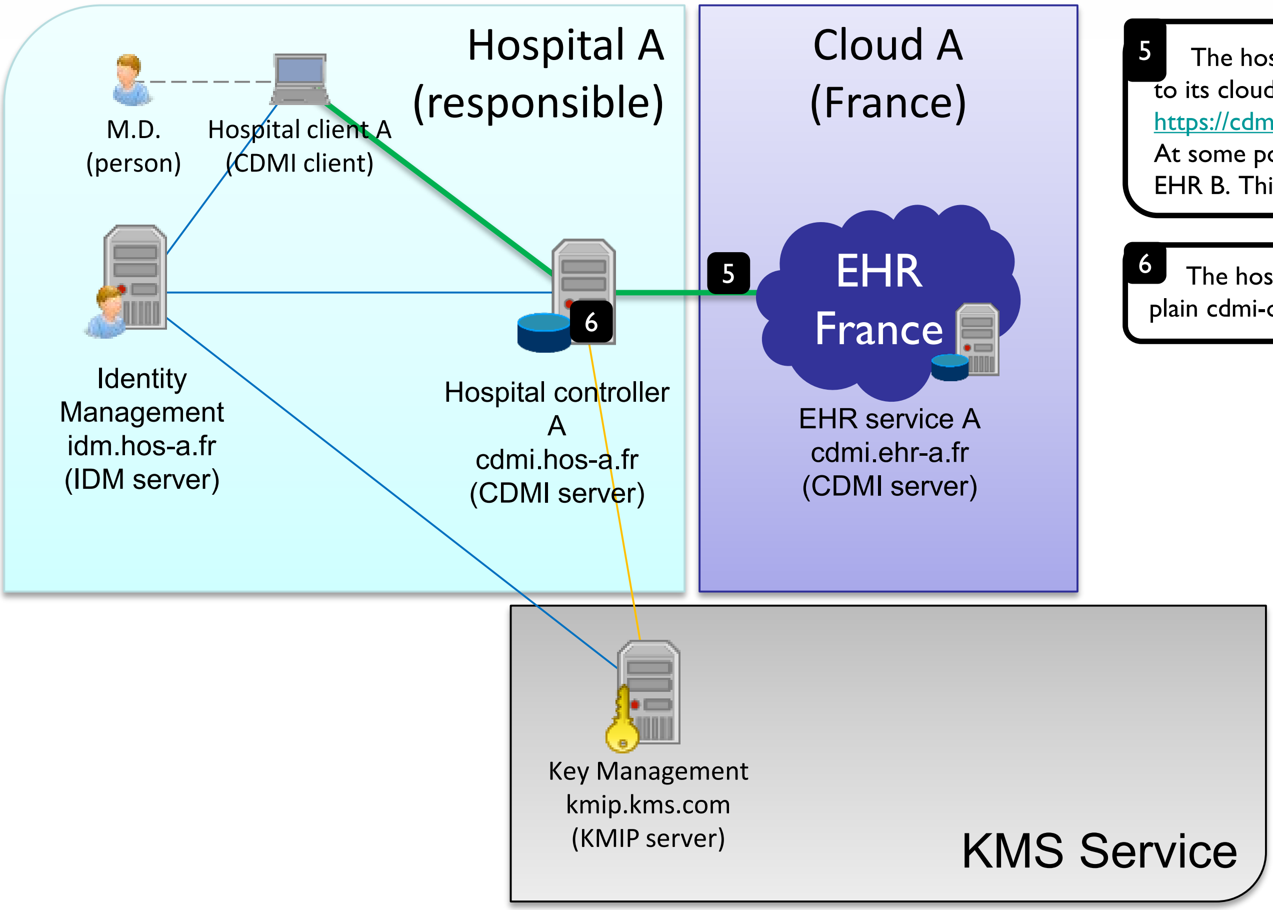
1 The M.D. in hospital A creates a new medical record and inputs it into the hospital client.

2 The hospital client contacts the hospital controller using CDMI

3 The hospital controller checks the authentication token against the IDM and receives ID information.
The IDM registers the storage of a new record in its audit logs.

4 The hospital controller does in-place modification of the CDMI-object

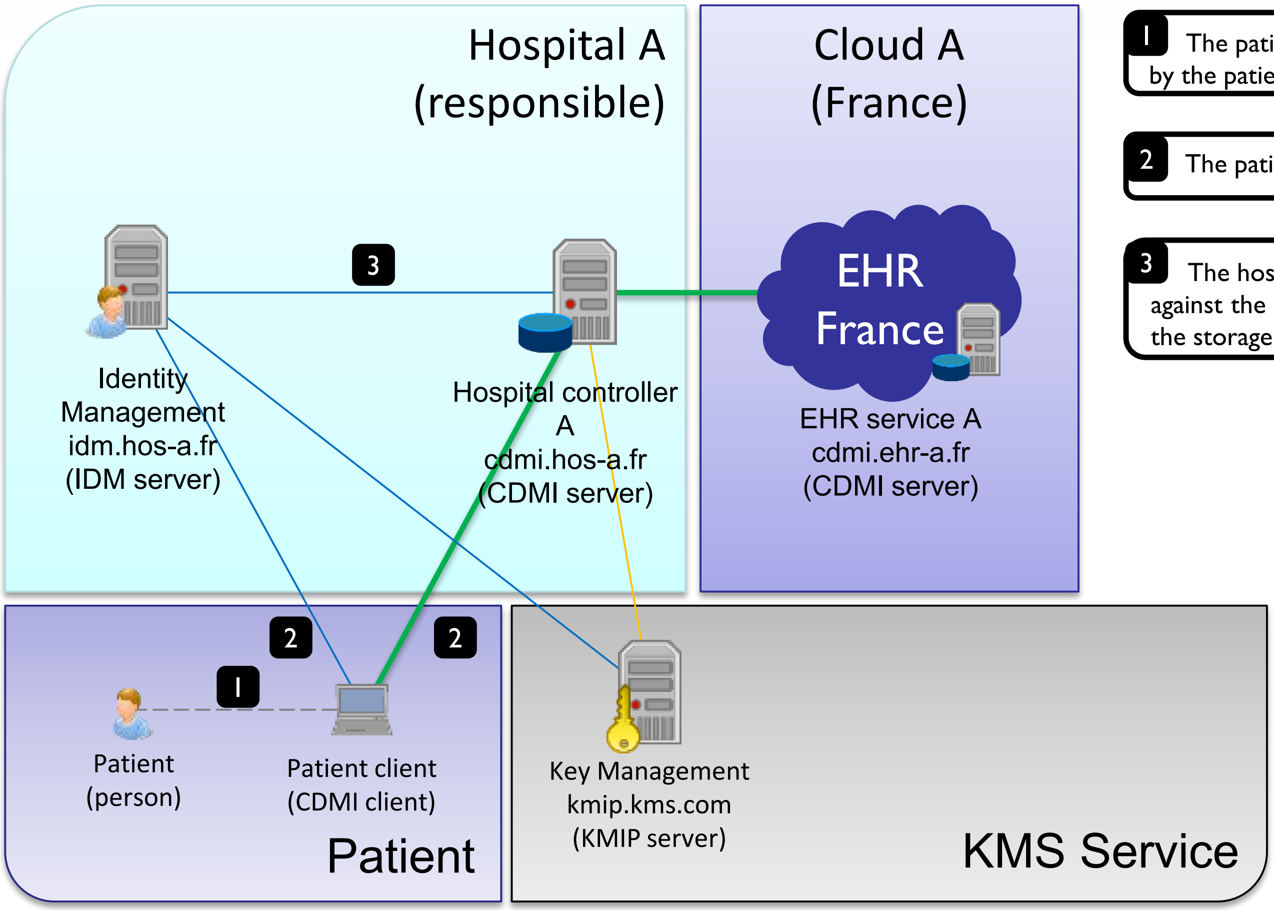
Scenario 1: Creation of Medical Record (2 of 2)



5 The hospital controller moves the encrypted CDMI-object to its cloud-based EHR A. This is a CDMI ‘PUT’ operation to https://cdmi.ehr-a.fr
At some point EHR A federates the encrypted CDMI-object to EHR B. This is a CDMI ‘PUT’ operation to https://cdmi.ehr-b.usa

6 The hospital controller deletes all cached data (content-key, plain cdmi-object, encrypted cdmi-object).

Scenario 2: Creation of Consent Profile (1 of 2)


Hospital A (responsible)
Cloud A (France)
3
EHR France
Identity Management idm.hos-a.fr (IDM server)
Hospital controller A cdmi.hos-a.fr (CDMI server)
EHR service A cdmi.ehr-a.fr (CDMI server)
2
2
1
Patient (person)
Patient client (CDMI client)
Patient
Key Management kmip.kms.com (KMIP server)
KMS Service

1 The patient creates a consent profile. This profile is digitalized by the patient client.

2 The patient client contacts the hospital controller using CDMI

3 The hospital controller checks the authentication token against the IDM and receives ID information. The IDM registers the storage of a new record in its audit logs.

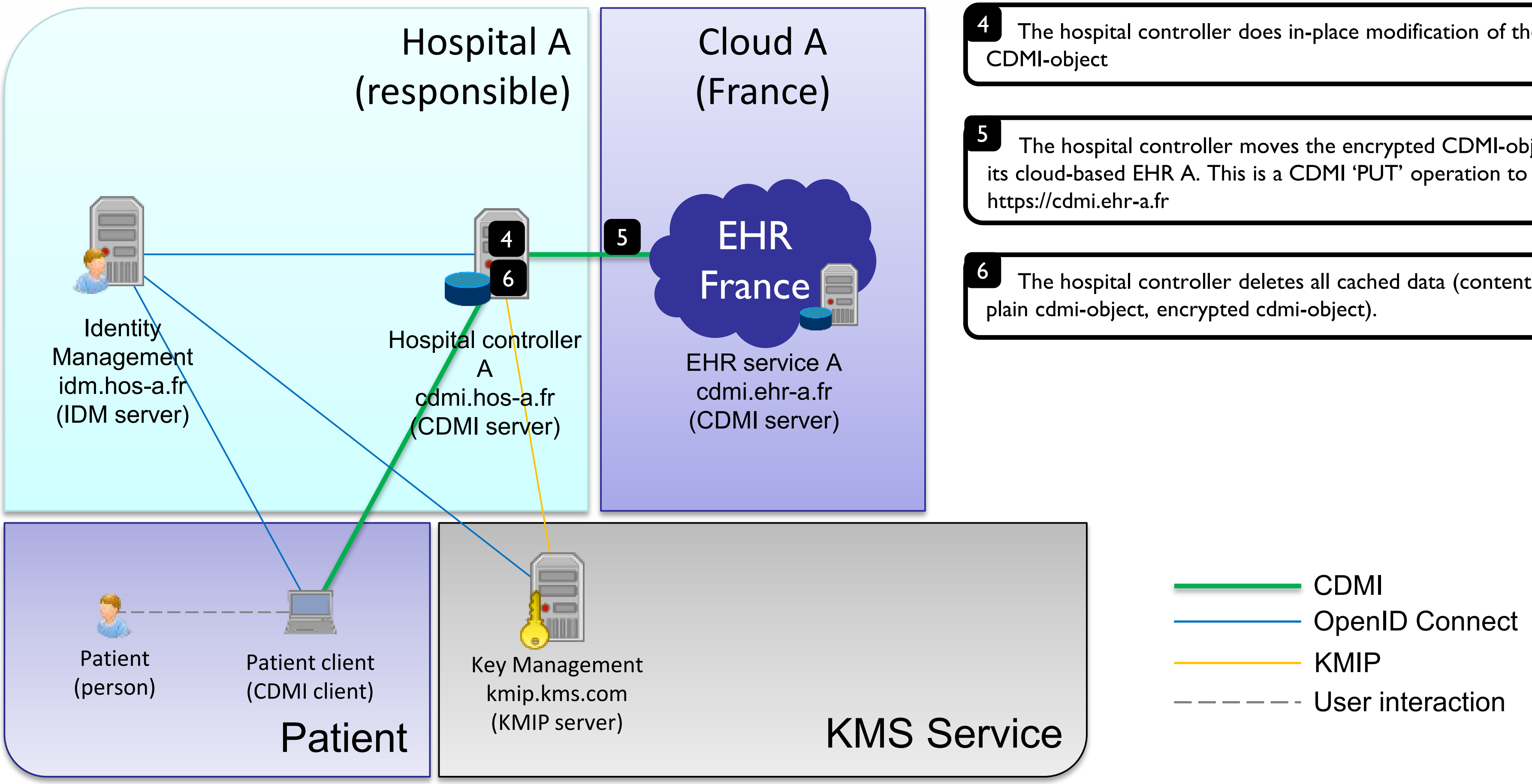
CDMI

OpenID Connect

KMIP

User interaction

Scenario 2: Creation of Consent Profile (2 of 2)

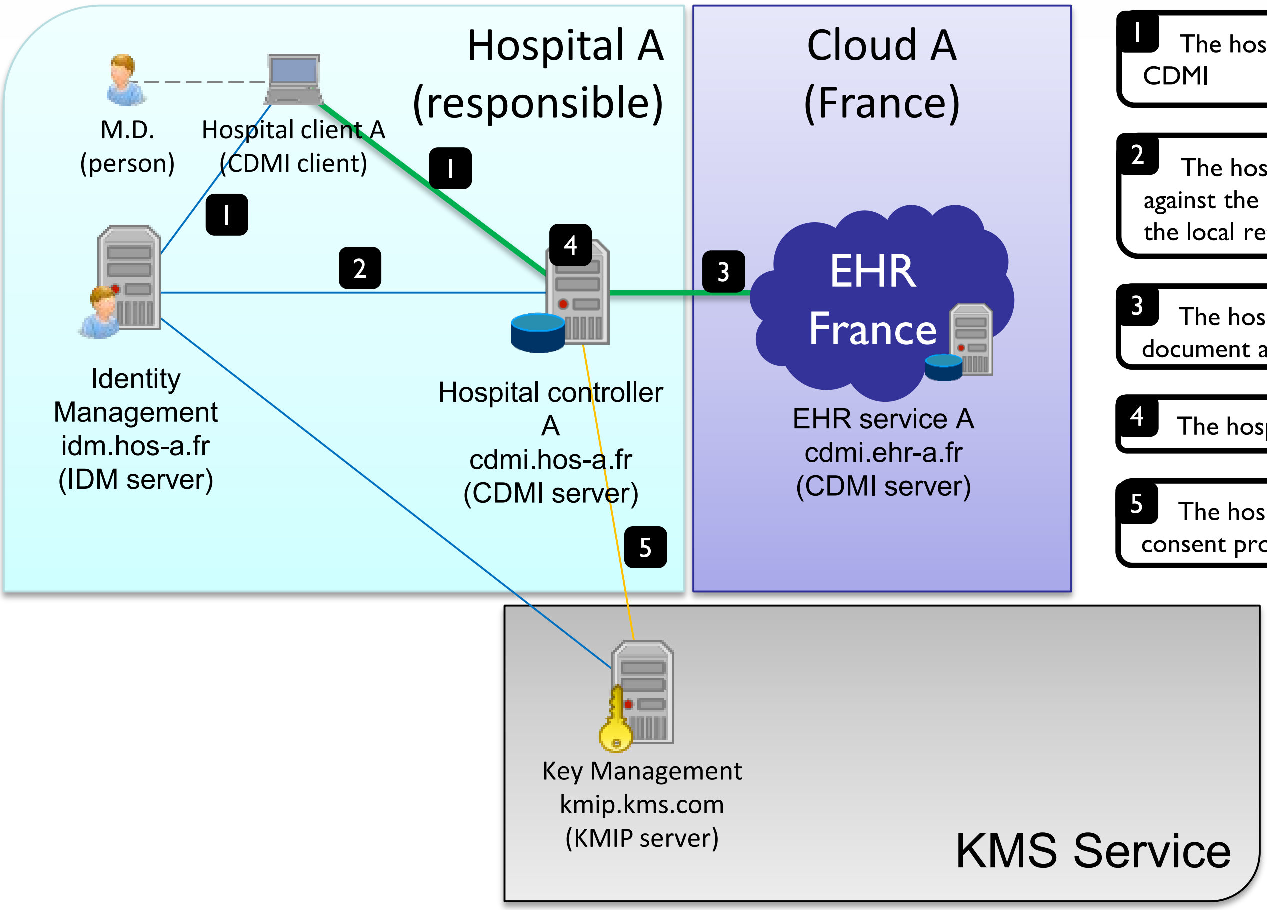

Hospital A (responsible)
Cloud A (France)
EHR France
Identity Management idm.hos-a.fr (IDM server)
4
6
5
Hospital controller A cdmi.hos-a.fr (CDMI server)
EHR service A cdmi.ehr-a.fr (CDMI server)
Patient (person)
Patient client (CDMI client)
Patient
Key Management kmip.kms.com (KMIP server)
KMS Service
CDMI
OpenID Connect
KMIP
User interaction

4 The hospital controller does in-place modification of the CDMI-object

5 The hospital controller moves the encrypted CDMI-object to its cloud-based EHR A. This is a CDMI ‘PUT’ operation to https://cdmi.ehr-a.fr

6 The hospital controller deletes all cached data (content-key, plain cdmi-object, encrypted cdmi-object).

Scenario 3: Local Request of Medical Record (1 of 3)



1 The hospital client contacts the hospital controller using CDMI

2 The hospital controller checks the authentication token against the IDM and receives ID information. The IDM registers the local retrieval of a record in its audit logs.

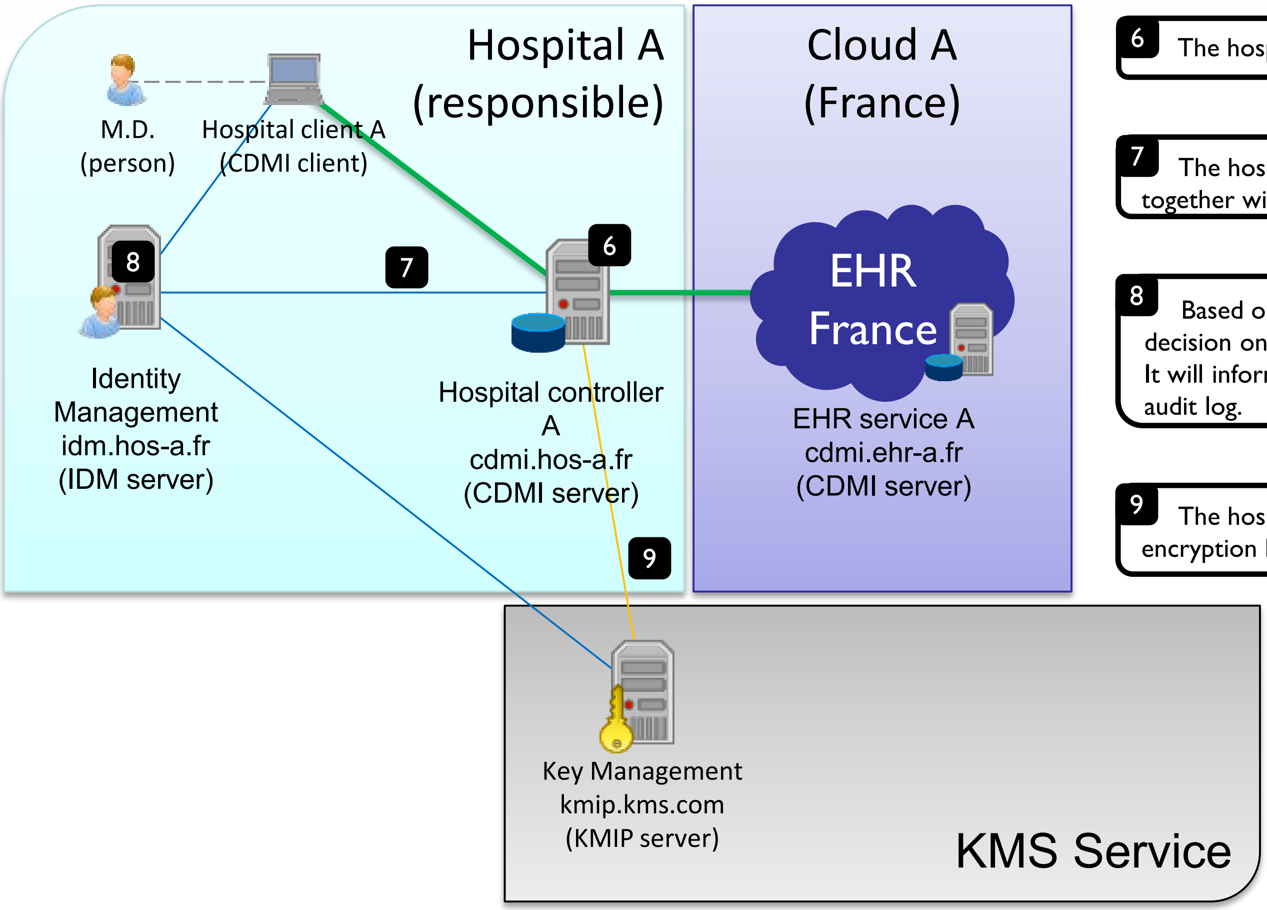
3 The hospital controller retrieves both the encrypted document and any encrypted consent profiles from EHR-A.

4 The hospital controller processes all objects.

5 The hospital controller obtains the encryption keys for the consent profiles from the KMS via the KMIP protocol

- ━━━ CDMI
- ─── OpenID Connect
- ─── KMIP
- – – – – – User interaction

Scenario 3: Local Request of Medical Record (2 of 3)



6 The hospital controller decrypts the consent profile.

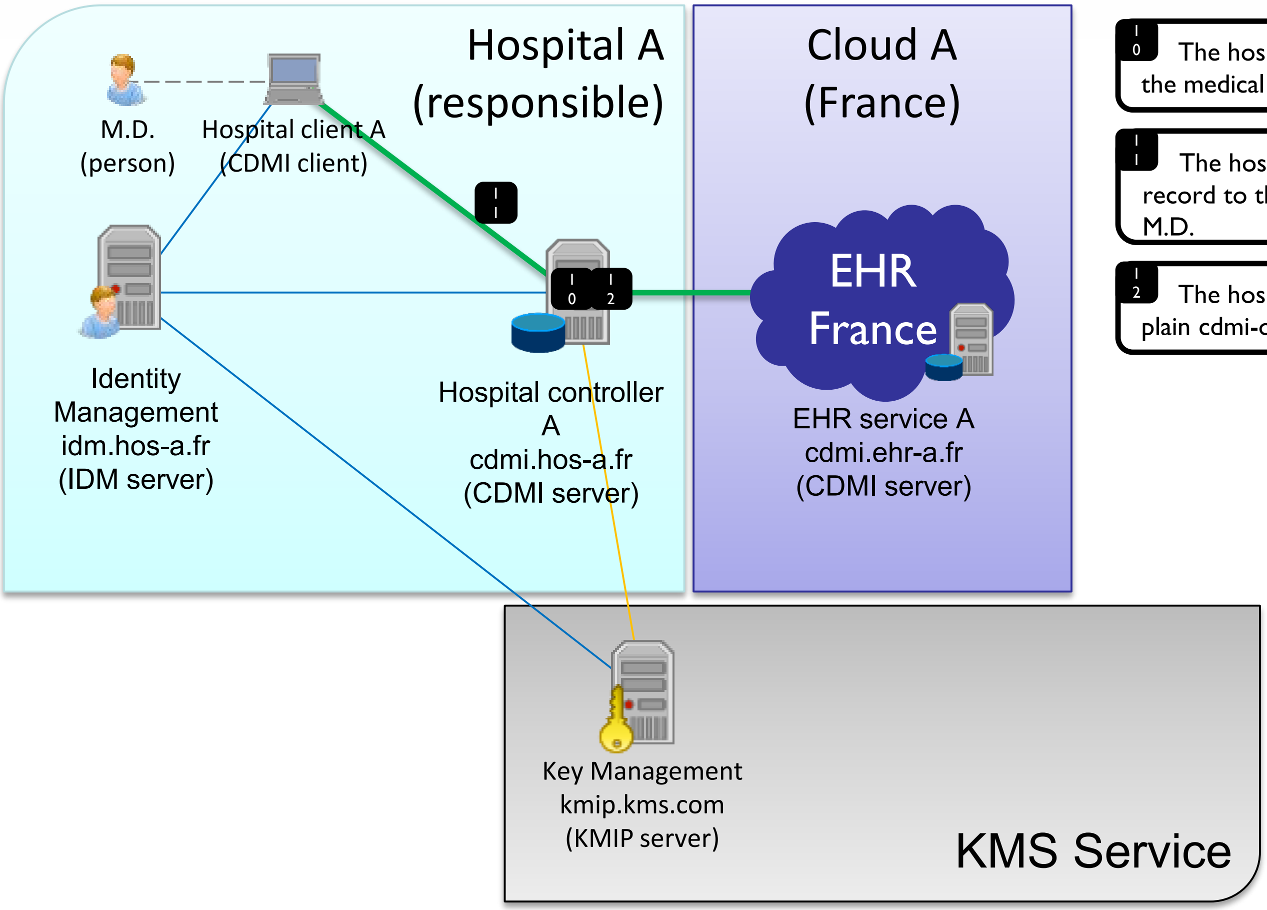
7 The hospital controller sends the consent profiles to the IDM, together with the M.D.'s authentication information.

8 Based on this information, the IDM will make an informed decision on whether the M.D. can have access or not.
It will inform the hospital controller and record an entry in its audit log.

9 The hospital controller will repeat step 5. to obtain encryption keys for the medical record.

━━━ CDMI
─── OpenID Connect
─── KMIP
– – – – – – User interaction

Scenario 3: Local Request of Medical Record (3 of 3)



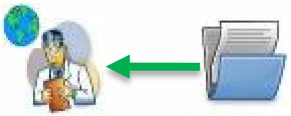
10 The hospital controller will perform in-place decryption of the medical record.

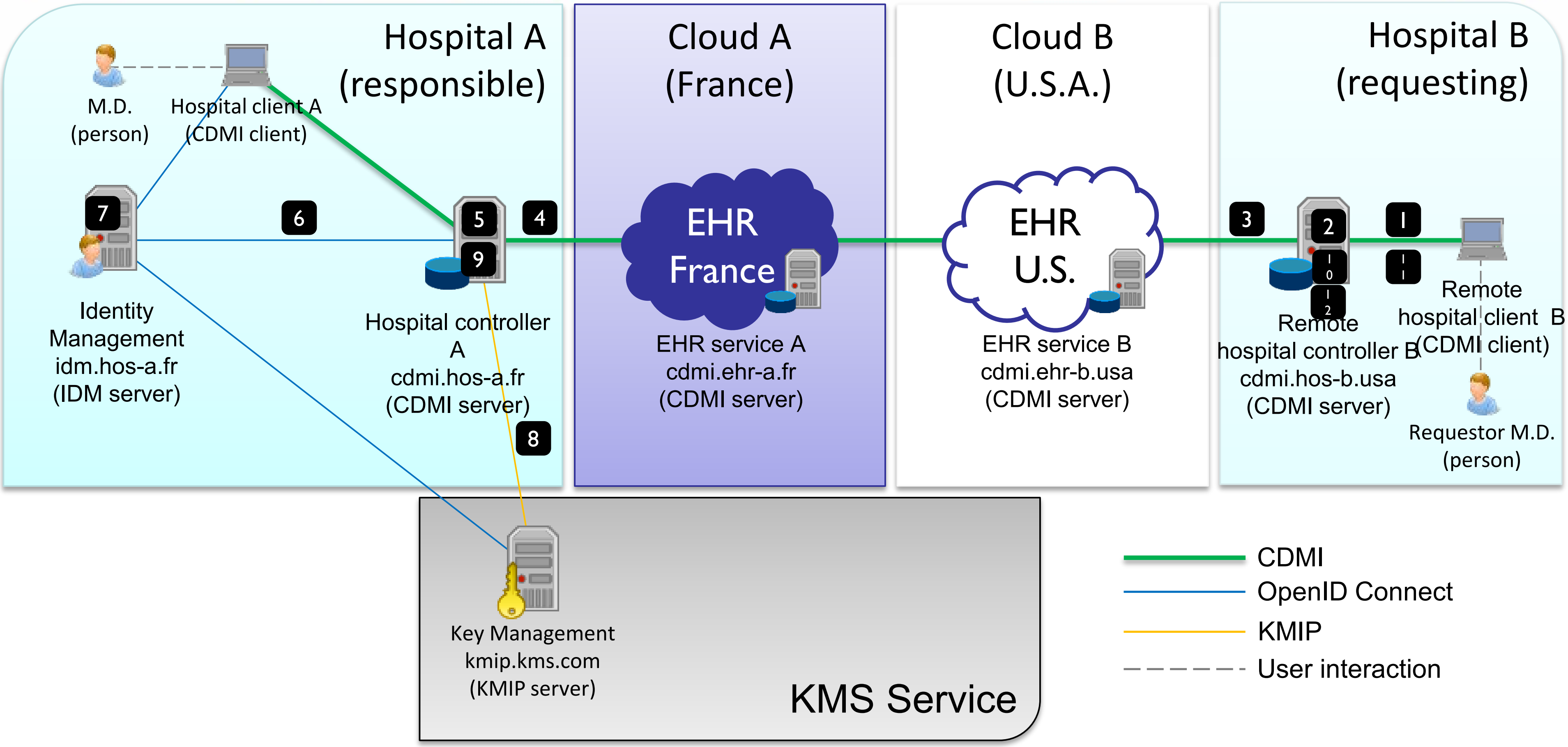
11 The hospital controller will send the decrypted medical record to the hospital client, where it can be accessed by the M.D.

12 The hospital controller deletes all cached data (content-keys, plain cdmi-objects, encrypted cdmi-objects).

- ▬ CDMI
- ▬ OpenID Connect
- ▬ KMIP
- – – – – – User interaction

Scenario 4: Remote Request of Medical Records (1 of 3)





Hospital A (responsible)

M.D. (person)

Hospital client A (CDMI client)

Identity Management idm.hos-a.fr (IDM server)

Hospital controller A cdmi.hos-a.fr (CDMI server)

Cloud A (France)

EHR France

EHR service A cdmi.ehr-a.fr (CDMI server)

Cloud B (U.S.A.)

EHR U.S.

EHR service B cdmi.ehr-b.usa (CDMI server)

Hospital B (requesting)

Remote hospital controller B cdmi.hos-b.usa (CDMI server)

Remote hospital client B (CDMI client)

Requestor M.D. (person)

Key Management kmip.kms.com (KMIP server)

KMS Service

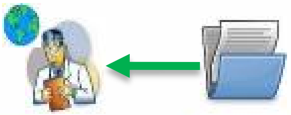
CDMI

OpenID Connect

KMIP

User interaction

Scenario 4: Remote Request of Medical Records (2 of 3)



1 The remote hospital client (B) contacts the remote hospital controller (B) using CDMI.

2 The remote hospital controller checks the authentication token against its local IDM (not displayed) and receives ID information of the requestor M.D..

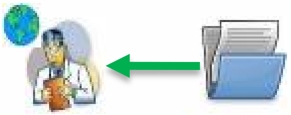
3 The remote hospital controller retrieves the encrypted document from EHR-B, which has a federated copy from EHR-A.

4 The remote hospital controller (B) creates and submits a CDMI DAC Request.

5 Hospital controller A processes the request.

6 Hospital controller A sends the consent profiles to its IDM, together with the decrypted DAC Request. This includes the fields 'client_identity' and 'X-DAC-AUTHORIZATION'.

Scenario 4: Remote Request of Medical Records (3 of 3)



7 Based on this information, the IDM makes an informed decision on whether the requestor M.D. can have access or not.

8 Hospital controller A uses this credential to obtain from the KMS encryption keys cdmi_enc_keyID for the medical record.

9 Hospital controller A prepares a CDMI DAC Response to return to hospital controller B.

10 The remote hospital controller B processes CDMI DAC Response.

11 The remote hospital controller B sends the decrypted medical record to the remote hospital client, where it can be accessed by the requesting M.D.

12 The remote hospital controller B deletes all cached data (content-key, plain cdmi-object, encrypted cdmi-object) as specified in the CDMI DAC response fields dac_key_cache, dac_response_cache_expiry, or as by prior agreement.

References

- Cloud Data Management Interface v1.1.1
  SNIA Technical Position, published March 19, 2015
- Towards a CDMI Healthcare Profile
  Rosner, Sedghi, Bernsen, Deng, Gu, published February 2015
- CDMI Encrypted Object Extension v1.1c
  SNIA Working Draft, November 19, 2015
- CDMI Delegated Access Control Extension v1.1b
  SNIA Working Draft, November 19, 2015

After This Webcast

- This webcast and a copy of the slides will be posted to the SNIA Cloud Storage Initiative website and available on-demand
  - http://www.snia.org/forum/csi/knowledge/webcasts
- A full Q&A from this webcast, including answers to questions we couldn't get to today, will be posted to the SNIA-CSI blog
  - http://www.sniacloud.com/
- Follow us on Twitter @SNIACloud

Conclusion

Thank You