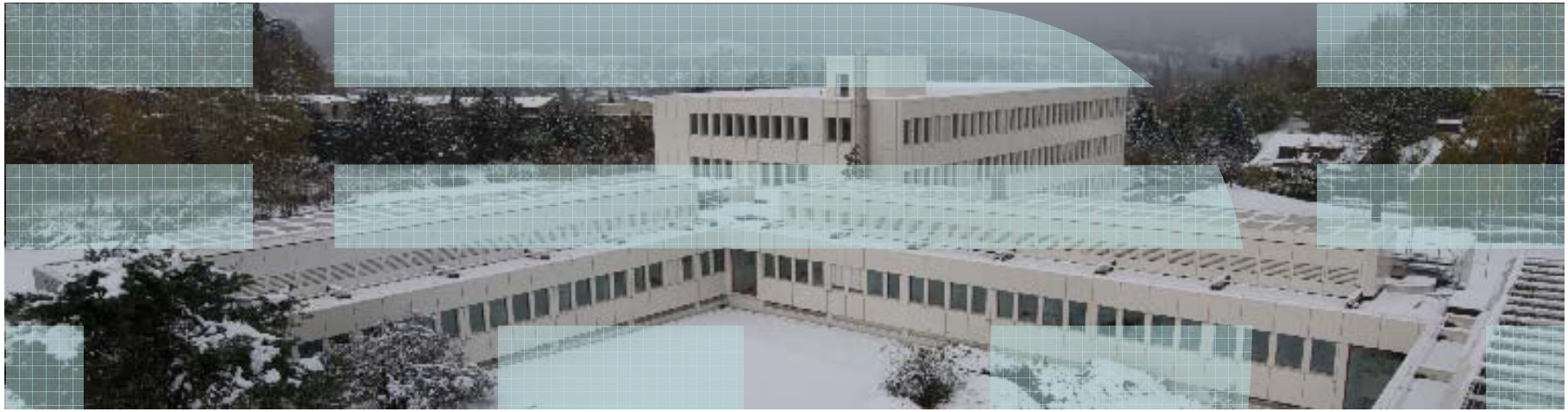
Robert Haas <rha@zurich.ibm.com>

September 28 2009

# Access Control in KMIPv2 (or v1.x)



# Access Control in KMIPv1

- Pretty much left out
- *Operation Policy Name* placeholder
  - Not mandatory
  - Defines the default permissions

# Access control in KMIPv2

- Should KMIPv2 specify access control mechanisms more precisely?
- What has to be exposed in KMIPv2 in order to properly implement AC?
- We identified some mechanisms to **strengthen** access control in future KMIP servers

# Access Control in KMIP

- Any KMIP server will need to implement access control policies
  - to distinguish between different users/roles
- Operation Policy Name in KMIPv1 suggests access control lists attached to each object
  - Operation Policy Name is an attribute of an object
- Basic access control for KMIP could include
  - ACLs attached to objects consisting of (user,permission) pairs
  - Global User Permission Lists (UPLs) for executing KMIP operations w/o existing object (e.g., Create, Register)
    - A UPL is a single, global list, not attached to any object containing (user, permission) pairs

# Users and Permissions

- Users
  - Any KMIP server will need to distinguish among users/roles
  - Special users *any* and *creator* already in KMIPv1

- (Proposed) Permissions on existing objects (for ACLs)
  - Admin (permits all operations),
  - Derive (permits key derivation using the key as a derivation key),
  - Destroy (permits destroy operation),
  - Get Wrapped (permits a key to be exported in a wrapped form),
  - Read (permits reading the key in cleartext, i.e, Get operation (unwrapped)),
  - ReadAttributes (permits gaining knowledge about key attributes),
  - Unwrap (permits a key to be used for unwrapping in the Register operation),
  - Wrap (permits a key to be used for wrapping in the Get operation)
  - …

- Global permissions (Permissions for UPL*s*)
  - Create
  - Register

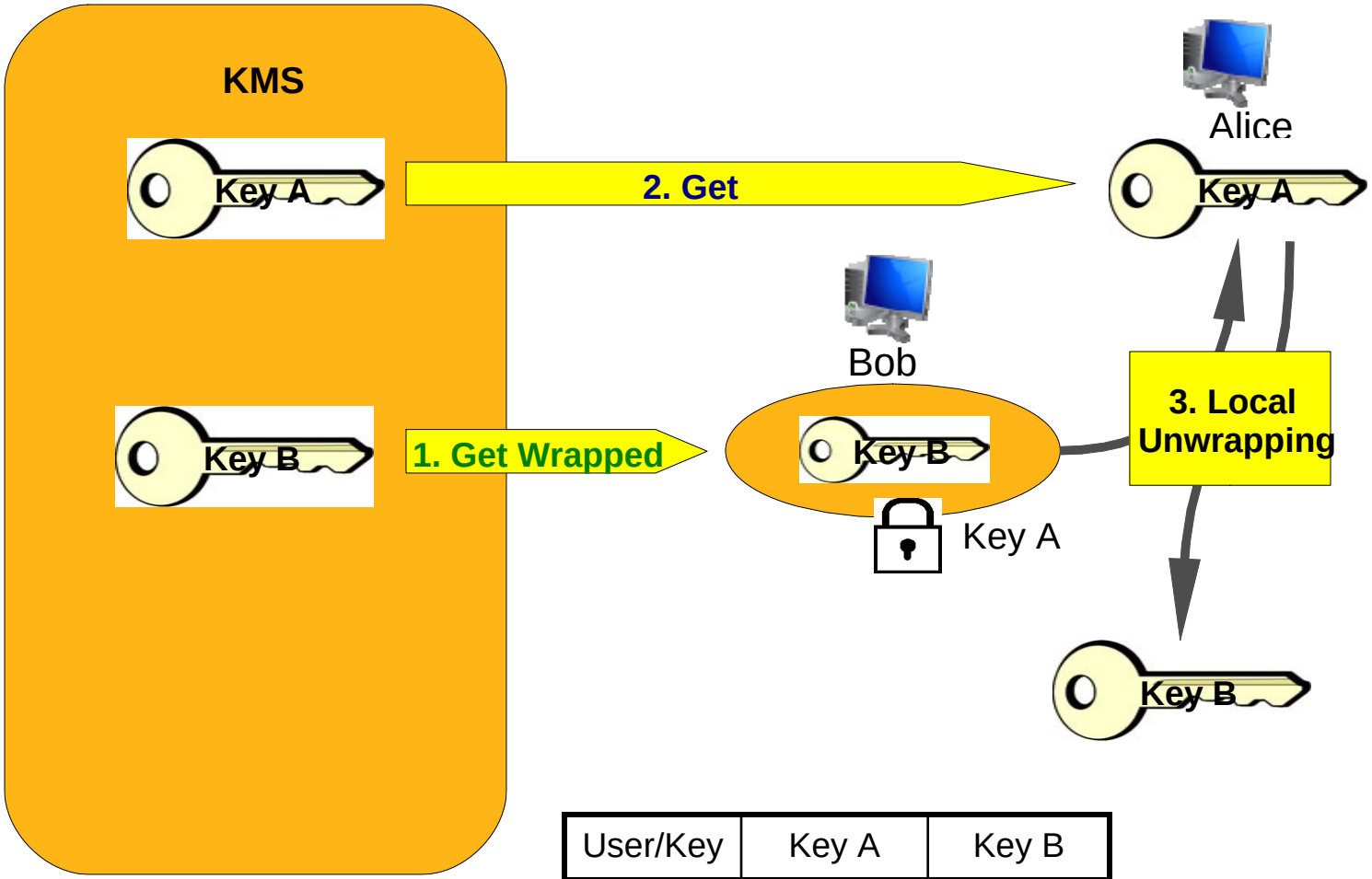
# One legitimate concern

- API attacks on key management APIs
- Operation on cryptographic keys might introduce dependencies between keys
  - Example operations: *Derive, Get* (wrapped)
  - If only basic ACL/RBAC mechanisms are implemented/specified in KMIPv2, these might be circumvented by exploiting the KMIP API
- Such API attacks are applicable to existing practical key management interfaces like PKCS #11

# Example of the attack

- Use case
  - Tape drive encryption
- Setup
  - Key B is used to encrypt the tape drive
  - Key B wrapped with key A is stored on a tape drive (e.g., by user Bob)
  - Initially no user (except administrator) has a permissions to read the key material of Key A
  - Administrator gives the permission to user Alice to read the key material on key A, not knowing that it was used to wrap key B
  - Administrator does not intend to allow Alice the permission to read key B
- Attack
  - Alice reads the key material of Key A from KMIP server, gets the wrapped Key B, unwraps it and obtains the key material of Key B

# Example (Get Wrapped)




| User/Key | Key A | Key B |
|---|---|---|
| Alice | Read | - |
| Bob | Wrap | Get Wrapped |

# API attacks

- Not really a concern for KMIPv1 since KMIPv1 does not define any mandatory access control mechanism
- But, they might be a concern in KMIPv2
  - A definition of cryptographically insecure access control mechanisms might be an issue

# Strict access control

- Proposed solution for the API attack problem in future KMIP servers
- Add custom attributes to KMIP to track dependencies among keys and modify basic ACL/RBAC mechanisms to account for these
  - Formal security policy
  - Security proof
- References:

1. Mathias Bjorkqvist, Christian Cachin, Robert Haas, Xiao-Yu Hu, Anil Kurmus, Rene Pawlitzek, and Marko Vukolic, *Design and Implementation of a Key-Lifecycle Management System,* IBM Research Report RZ 3739, June 2009.

2. Christian Cachin and Nishanth Chandran. A secure cryptographic token interface. In Proc. Computer Security Foundations Symposium (CSF-22). IEEE, July 2009.

# Other items on access control agenda for KMIPv2?

- Suggestions?