Marko Vukolic <mvu@zurich.ibm.com>

December 9, 2009

Server-to-server (s2s) in KMIPv1.1/v2

Outline

- Server-to-server (s2s) use cases (*presented at the KMIP f2f meeting*) and key functionalities needed
- Deficiencies of KMIPv1 for s2s
- Summary and possible actions in KMIPv1.1./v2

Server to server (s2s) use cases

- Backup, Data Loss Prevention
- Load balancing/Delegation
- Propagating key material closer to endpoints, e.g.,
  - Example 1 (A retail store)
    - A retail store operation with each store relying on encrypted storage
    - network connectivity with the central key management server (CKMS) not reliable
    - small subset of the keys needed to be served locally, but the management is at CKMS
    - Keys at local key-management servers could be read-only, with pre-allocated usage or lease time
    - The local server needs to communicate with the CKMS
  - Example 2 (e-commerce websites)
    - Multiple e-commerce websites centrally managed (CKMS)
    - Some keys need to be pushed down from CKMS (readable locally), i.e., with CKMS exporting the keys
- Propagating key material updates towards the central key manager
  - A large multinational bank needs the information about cryptographic material from Location B in central Location A (but not vice versa)

Server to server (s2s) use cases (cnt'd)

- Business-partner data exchange
  - Propagation of keys between KMIP servers to facilitate business partner data exchange
- Partitioning and M&A
  - A KMIP server needs to be partitioned into more servers
  - A company acquires another and cryptographic objects from different KMIP servers need to be merged
- KMIP server acting as the gateway/proxy
  - A less capable KMIP server may need to proxy client's request to the more capable KMIP server (e.g., to interact with a PKI)
- Replication (fault-tolerance)
- Exchange of different server policies and their enforcement (*not the focus of this presentation*)

Additional use cases?

- Suggestions?

Needed functionality (high-level overview)

- Backup, Data Loss Prevention
  - Bulk export
  - Additional object attributes
    - Backup flag: denoting that the backup server holds the read-only, backup copy.
    - The representation of the working server (from which backup was initiated)

- Load balancing/Delegation, Propagating key material closer to endpoints,
  - Bulk export/import
  - Additional object attributes
    - Master (key owner) – roughly: this is the KMIP server is responsible for handling updates on the object metadata (TBD: single or multiple valued Master attribute)

- Propagating key material updates towards the central key server
  - Bulk import
  - Attribute synchronization

Needed functionality (cnt'd)

- Business-partner data exchange
  - Bulk export/import
- Partitioning and M&A
  - Bulk export/import
  - Master attribute
- KMIP server acting as the gateway/proxy
  - Use KMIP v1
  - Proxying clients' credentials
- Replication (fault-tolerance)
  - Bulk export/import (for initial load)
  - Attribute synchronization (for updates)

(Some) key functionalities needed for s2s use cases

- Bulk export/import support
- Attribute synchronization
- Additional attributes

Outline

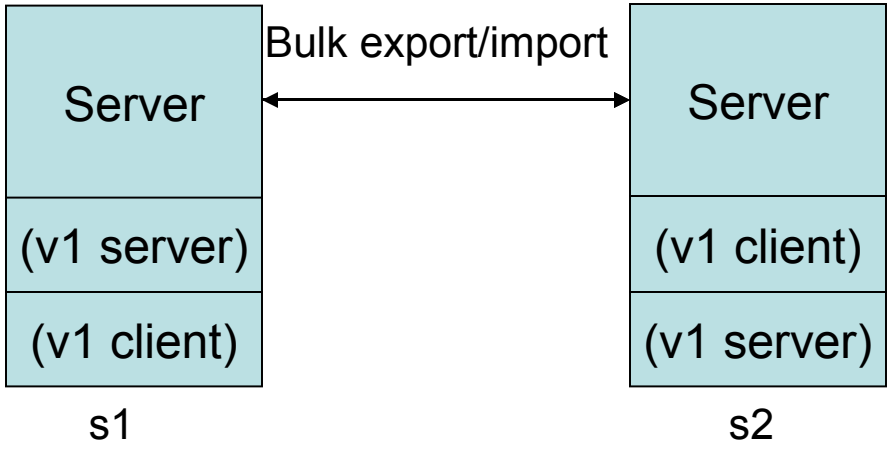
- Server-to-server (s2s) use cases (*presented at the KMIP f2f meeting*) and key functionalities needed
- Deficiencies of KMIPv1 for s2s
- Summary and possible actions in KMIPv1.1./v2

Deficiencies of KMIPv1 for s2s

How about using KMIPv1 to emulate bulk export/import and attribute synchronization?

1. Bulk export/import with KMIPv1

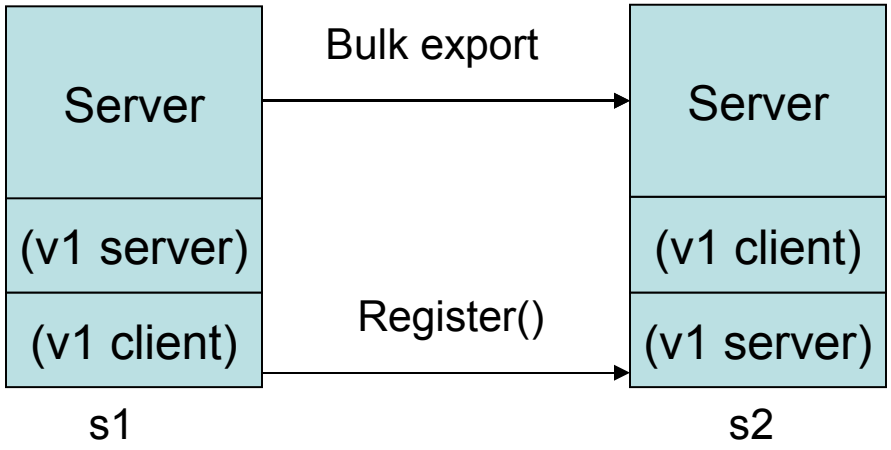
- Goal: Export/import a set of objects (possibly all) contained in KMIP server s1 to/in server s2 (using KMIPv1)



- We identified 3 different ways to achieve this in KMIPv1:
  - Each has its shortcomings

1a. Bulk export using Register

- At s1:
  Forall objects obj_i:
  s2.register(obj_i)



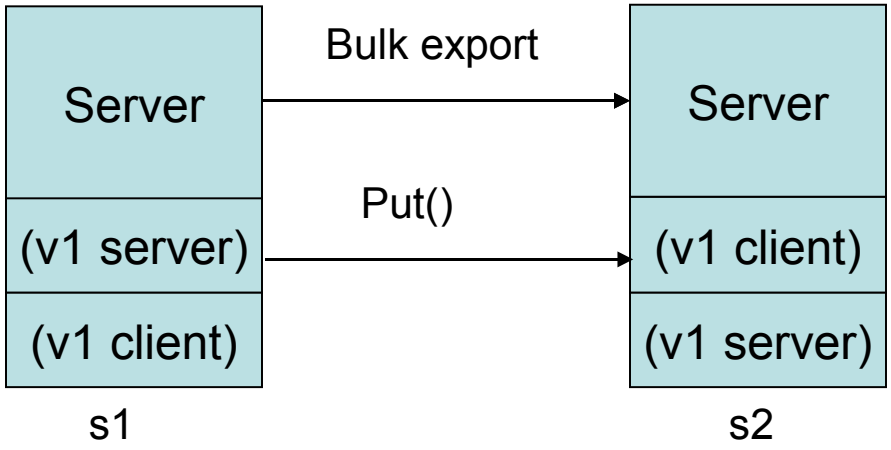
- s1 lists all objects locally and registers them at s2 (acting as a client)
- Issues:
  - per object registration (batching may help)
  - KMIPv1 client (s1) cannot enforce all attributes to be set as it wishes. The following attributes cannot be enforced, e.g.:
    - Unique Identifier, Digest, Lease Time, Initial Date, Destroy, Compromise Occurrence Date, Compromise Date, Revocation Reason, Link, Last Changed Date,…
    - Other attributes set initially by the server might pose problems as well
  - UUID, Name collisions with objects already existing at s2

1b. Bulk export using Put

- At s1:

  Forall objects obj_i:

  Put(s2,obj_i)

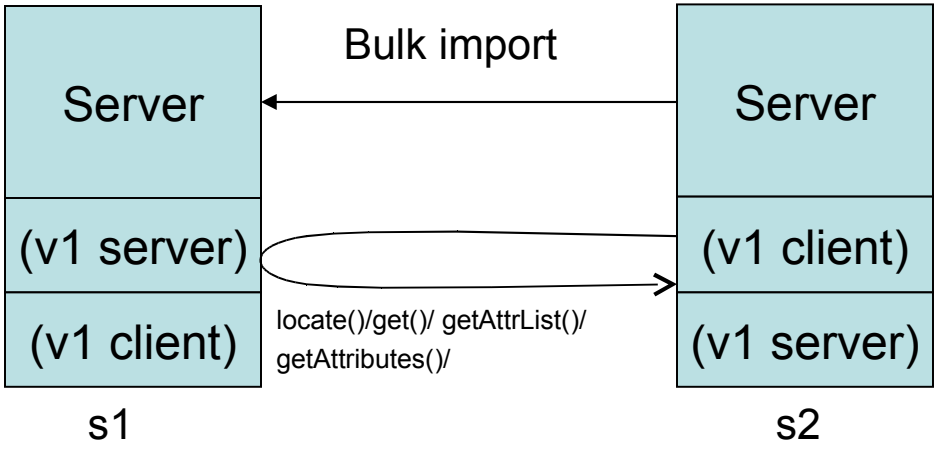


- s1 lists all objects locally and puts them to s2 (acting as a KMIPv1 server)
- Issues:
  - per object Put (batching may help)
  - Put is optional in KMIPv1 (limits interoperability)
  - UUID, Name collisions with objects already existing at s2

1c. Bulk import using Locate/Get/Get Attribute List/Get Attributes

- At s2:

```
s1.Locate(*)
Forall uuid[i] {
    s1.Get(uuid[i])
    s1.GetAttrList(uuid[i])
    s1.GetAttrs(uuid[i],*)
}
```



- s2 locates all keys at s1 and gets them one by one
- Issues:
  - Slow: per object retrieval (batching seems not to help here since GetAttrList and GetAttrs cannot be batched); to get object with all attributes need 3 ops per object;
  - s2 might limit the number of responses to Locate*, s1 would then need to come up with a good strategy to extract all objects, e.g., by segmenting the namespace. Not trivial and even slower.
  - Cannot “Locate All” in KMIP v1 (cf. s1.Locate(*))

*Indication to the client that this limit is reached is missing from KMIP v1

2. Attribute synchronization in KMIP v1 (very briefly)

We explored 2 possibilities (assuming a single master server per object, represented by Master attribute) – details omitted

1) Publish-subscribe like synchronization
   - Where Master maintains a list of slave servers (in a separate attribute Slave) + Notify operation (or Put with Replaced Unique Identifier (RUI) specified)
   - On update (change of state) of an object, the Master informs all slaves about the update using Notify/Put (w. RUI specified)

2) On demand synchronization
   - Similar to bulk import, i.e., using Locate issued to the Master server, with Last Change Date as a parameter of Locate

- Issues:
  - Notify is optional
  - There is no way to notify the slave of attribute deletion using KMIPv1 Notify
  - The behavior of Put when Replaced Unique Identifier ruuid is specified but the object with ruuid does not exist, is not specified by KMIPv1
  - Useful Custom attributes should be standardized and attributes made mandatory
  - Locate issues (cf. Bulk import)

Outline

- Server-to-server (s2s) use cases (*presented at the KMIP f2f meeting*) and key functionalities needed
- Deficiencies of KMIPv1 for s2s
- Summary and possible actions in KMIPv1.1./v2

Shortcomings of KMIPv1 and possible actions in KMIP v1.1./v2

- Useful operations are optional (Notify, Put)
  - KMIPv1.1/v2: make Notify and Put mandatory for a s2s compliant KMIP server
- Bulk export/import can be only partially emulated (using batched operations)
  - KMIPv1.1/v2: Need for “Get All Attributes” operation (this can be simply batched with Get to get an entire object and all its attributes)
- The behavior of Put when Replaced Unique Identifier ruuid is specified, but the object with ruuid does not exist on the remote end needs to be specified
  - KMIPv1.1/v2: Specify the behavior in the above case (**this could be preferably done even in KMIPv1**). One option is to require the server on the remote end to create an object and not fail the operation
- Notify does not support notification about deleted attributes
  - KMIPv1.1/v2: Augment Notify to support attribute deletion notification (**this could be preferably done even in KMIPv1**).
- KMIPv1.1/v2: More attributes are needed (e.g., Master, Slaves, Backup flag, Working server (denoting the origin of the backed-up objects))
  - Single-valued Master would lead to avoiding complex conflict resolution and reconciliation mechanisms
  - Slaves could be able to modify the state of their local copy in some operations, e.g., Get Usage Allocation (details and per-operation analysis omitted from this presentation)
- Other issues
  - UUID, Name collisions across different servers
  - Locate does not return an indication to the client whether there are more objects matching the query (nor the means to “resume” such a Locate)
  - Cannot Locate all

Additional issues revealed that could/should be fixed in KMIPv1

- Besides those related to Put/Notify (see previous slide)
- Locate supports wildcards only for Name and Object group
  - This should be extended to other attributes represented as Strings (e.g., Application Specific ID)
- Not possible to Locate All objects in KMIPv1
  - Could be done simply by, for example, omitting all attributes in a Locate request
- These could be included into changes resulting from the Public Review

Comments?