Group Membership Service: Specification and Considerations

Brian Major

May 2017

IVOA Interop Shanghai

National Research Council Canada
Conseil national de recherches Canada
Canada

Agenda

- Quick review of GMS concepts (from Trieste Interop Meeting)
- Groups
- GMS and CDP (Credential Delegation Protocol)
- Users and GMS
- GMS Patterns
- API Definition
- Implementation
- Discussion

Overview

Group Management / Membership Service

A web service with a RESTful API that allows for the determination of whether a user is a member of a group.

The answer to this ‘isMember’ question can be use to allow (or deny) the user access to a resource.

This is the ***authorization decision***.

The owner(s) of the resource may, at any time, change the group and/or the group memberships of the groups that is protecting the resource.

This is the ***granting and revoking*** of access**.**

Resources

Resources are entities that may require authorization for access.

For example:

- Services (SIA, SODA, Processing, HR System)
- Data (Archive data, VOSpace files, TAP rows, catalogues, etc…)
- Metadata (TAP tables, VOSpace Node metadata, etc…)

Authorization Requirements

1) To allow for restricted access certain resources

Only a certain set of individuals may access certain resources

2) To allow certain individuals to set the access rules on resources

The owner(s) of the resources need to manage the access rules

Interoperable Authorization Requirements

3) To be able to re-use granting rules between resources

Projects must authorize access to a variety of proprietary resources

4) To be able manage granting rules at a single location

Projects should not have to update each resource on a change to a re-used grant

5) To be able to reference remote granting rules

Proprietary resources should not be confined to a single institution


Groups

Granting rules: groups

A single individual is too restrictive

Having a list of individuals is difficult to maintain

Grouping individuals and referencing them by a group identifier provides a necessary level of abstraction

Data Model

User <--> Group

User <--> Membership <--> Group

Membership object?

- Could contain extra information about the type of membership
- For example: “administrator” or “student”

Concept of “Roles” can be achived through group-membership alone.

Universal Group Identifiers

```
ivo://authority.example.com/gms?groupName
```

To resolve the host GMS service, lookup, in the Registry, the URL for serviceID:

```
ivo://authority.example.com/gms
```

This may result in (for example):

```
http://server.example.com/myGMS
```

Where to maintain the group grants?

There are two main options:

1) A centralized DB that maps resourceID to groupID.
2) Put the groupID with the resource.

- option #1 is not scalable.
- option #2 scales with resources.
- option #2 is interoperable between authorities

So, #2.

For example, the metadata (properites) of a VOSpace Node:

```
isPublic=false
groupRead=ivo://authority.com/gms?nodeReadGroup
```

List of Groups or Groups of Groups?

- The resource contains the authorized group (or groups)
- If the GMS service supports the concept of “Groups within Groups”, then:
  - The combined membership of a list of groups can be represented by a single group with groups within.
  - Thus, the resource only needs to reference one group.
  - The maintenance of the group membership becomes simpler.
- Seems good, however… what if the list of groups is spread across multiple authorities (institues)?
  - “Groups within groups” is more difficult to implement, but is possible (transitive CDP)

GMS and CDP

Why CDP? Reason 1: Information Privacy

Why make Service calls on behalf of the user?

Likely user data privacy policies:

1. User and user membership data is not public
2. Only the user is allowed to see their membership information.

So: Only the user can make a GMS isMember() call

How do services ***be*** the user for the isMember() call?
Answer: Use the credentials they have delegated through CDP

GMS and Credential Delegation Protocol

- Calls are ALWAYS made by the user trying to access the resource
- This is achieved by services using the users' (delegated) credentials
- The user does not know if his/her credentials will be required to make a secondary call
- Furthermore, the service does not know if the secondary call will require the user's credentials to make another
- Thus, there is a general preconditions for making calls in the IVOA Service-Oriented architecure: always delegate credentials before making the service call.
- The credentials can be short lived
- Services can delegate credentials on behalf of the user.

Discovering the right CDP service

Options:

- TAPRegEx: find all CDP endpoints and match on the authority of the GMS service?

- Is it possible to put the CDP standardID inside the GMS capability document?

Delegation of Trust

- Since the user's credentials can be distributed to any of the cooperating institutions in an interoperable IVOA network, the user inherently must trust each of these institutions
- When user accounts are created at the home institution, it should be made clear that

CDP for other types of credentials

- CDP enables single sign-on (SSO) in a SOA (service oriented architecture.

- Current document addresses X.509 client proxy certificates only
- They must be short-lived
- Add support for a token based approach?
- OAuth Cairvoyant API?
- Shibboleth?


Users

User Identities

- GMS must know:
  - The type of identity the user has provided
  - The value of that identity (the userID).
- The identity can be local or global:
  - Local: A userid in the institute
  - Global:—the identifying authority must be stated
    - X.509 client certificates – the user's Distinguished Name
      - Technically eligant: certificate IS the identity, no user info needs to be persisted anywhere
    - OAuth 2.0 tokens – TBD: more info needed on oauth user identifiers
- Institutions don't necessarily need to save users' identities.
  - Read access to resources can be maintained by remote GMS groups
  - Saving is only needed when they become owners of local resources
  - Should use internal identity abstraction to avoid maintenance on ID changes

User ID Type

{idType} defines the type of userID.

From SSO 2.0:

SSO mechanism	***<securityMethod>***
HTTP Basic Authentication	*ivo://ivoa.net/sso#BasicAA*
TLS with password	*ivo://ivoa.net/sso#tls-with-password*
TLS with client certificate	*ivo://ivoa.net/sso#tls-with-certificate*
Cookies	*ivo://ivoa.net/sso#cookie*
Open Authentication	*ivo://ivoa.net/sso#OAuth*
SAML	*ivo://ivoa.net/sso#saml2.0*
OpenID	*ivo://ivoa.net/sso#OpenID*

GMS Patterns

A general three step CDP pattern in an IVOA Service Oriented Architecture

1. Lookup desired service capabilities through the registry
2. Using the CDP endpoint provided in the capabilities, delegate the user's credentials
3. Make the service call

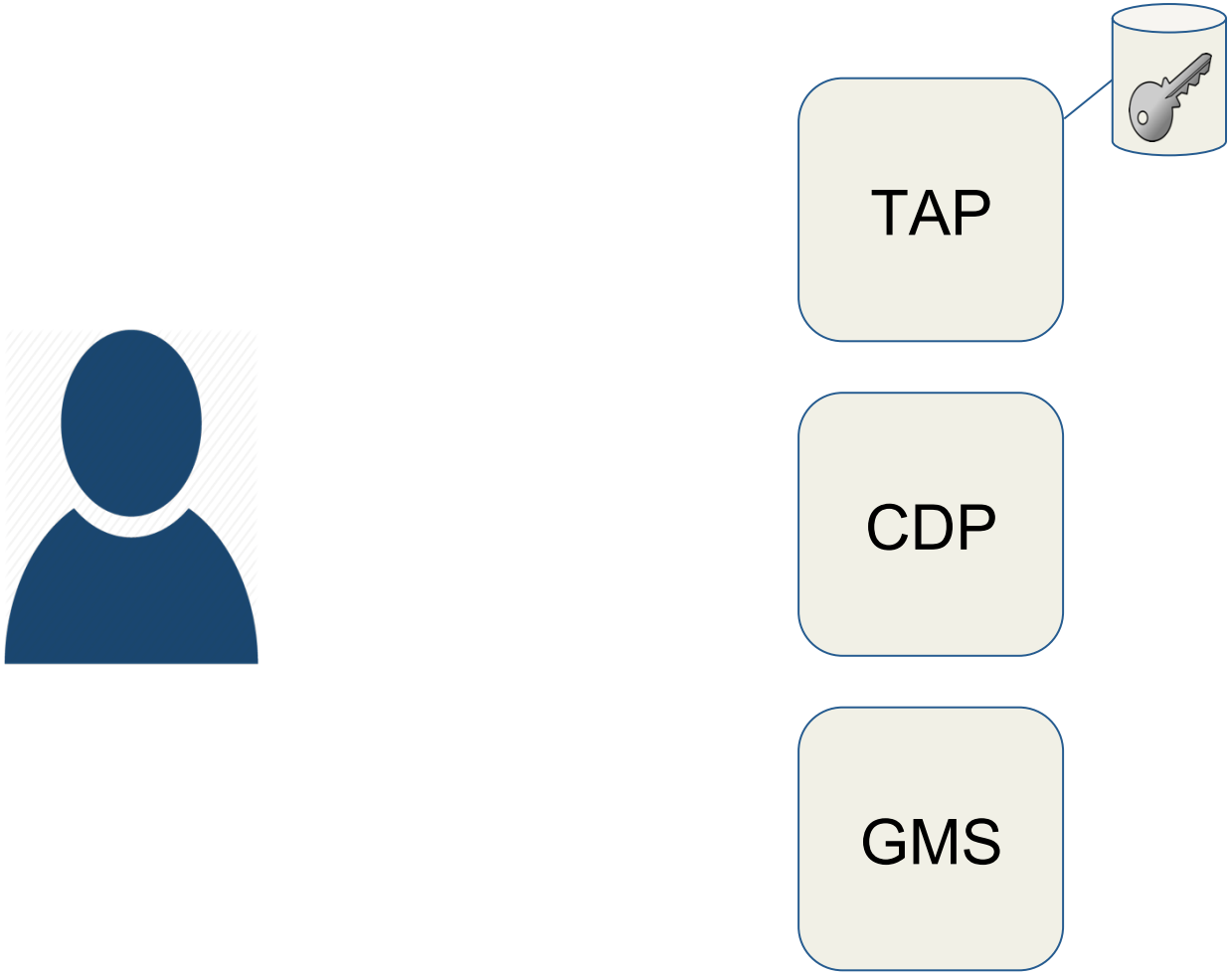
For both:

1. Initial calls made directly by the user
2. Any secondary calls make by services on behalf of the user

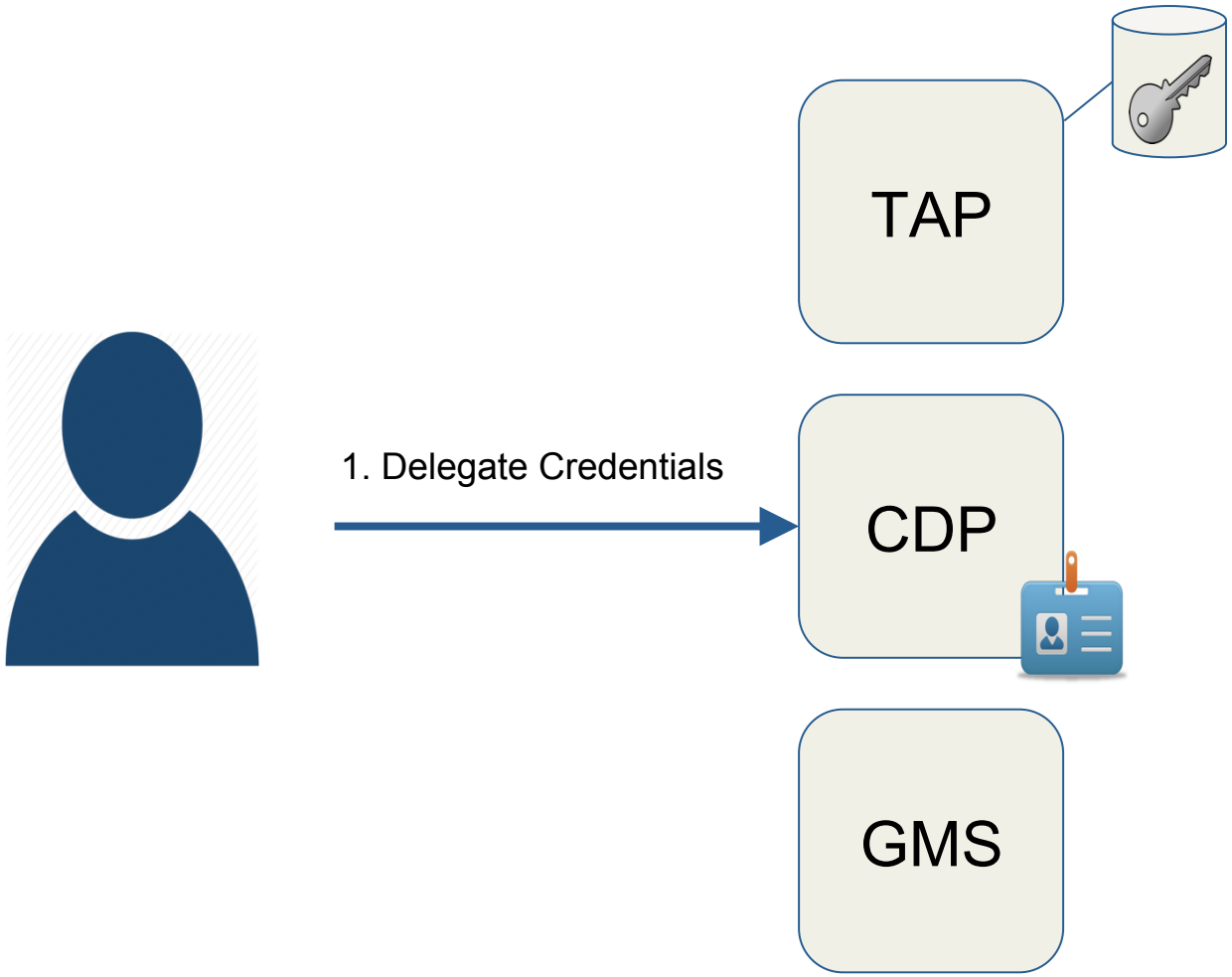
Details of this pattern for GMS isMember() call.

1. Get the identity of the user making the call
2. See what authentication types (SecurityMethods) the GMS service supports (via a capabilities call)
3. If the user's identity doesn't match one of the supported SecurityMethods, see if you have other identity types for that user. (For example, see if the userid/password account has an associated X.509 Distinguished name)
4. Is the identity is already a delegatable credential (such as an OAuth token)? If not, use the private API of your local CDP to get the user's delegated credential.
5. Delegate this credential to the remote GMS if it is not part of the local CDP authority.
6. Make the remote GMS isMember() with the user's delegated credential.

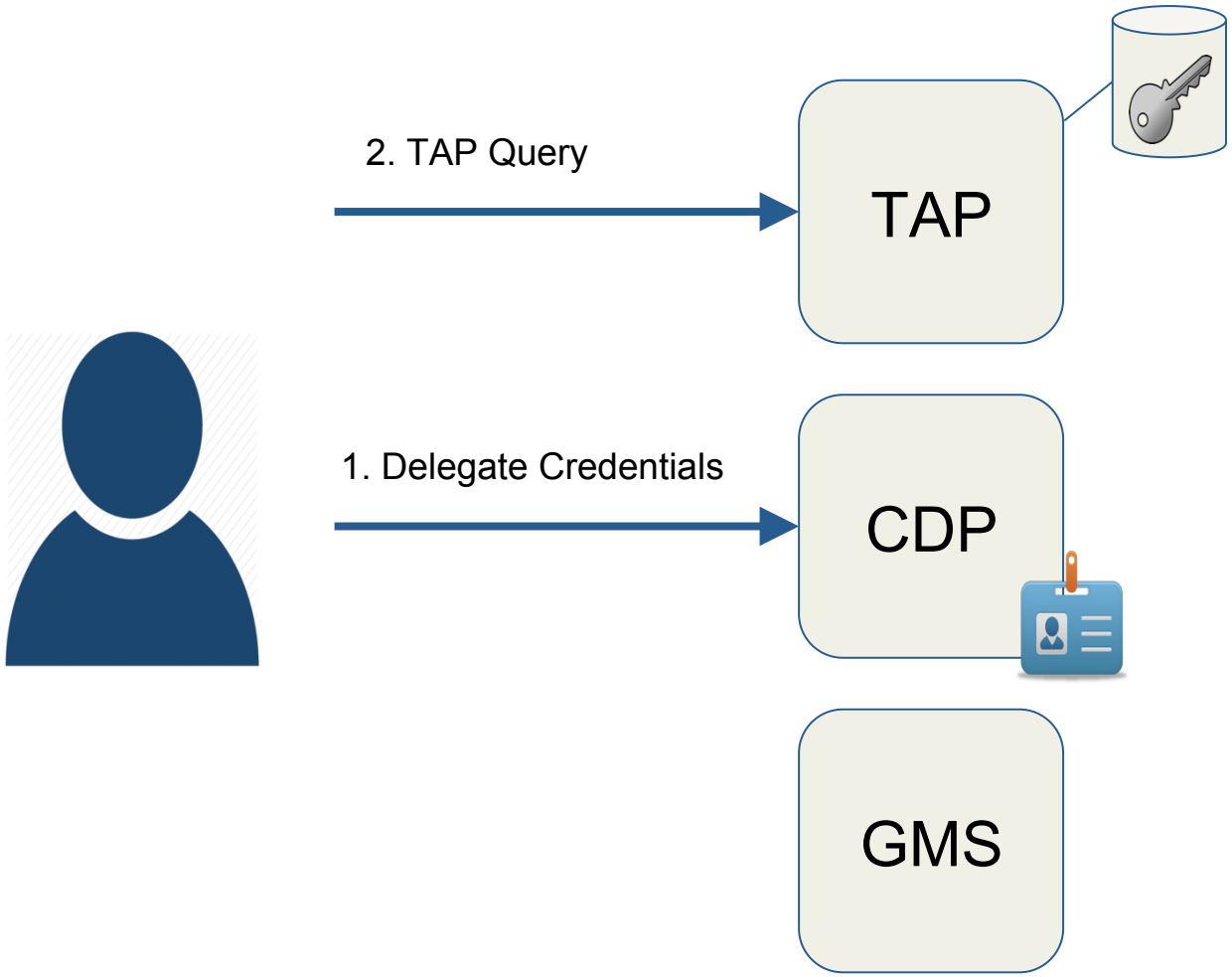
Scenario 1: Authorized TAP Query



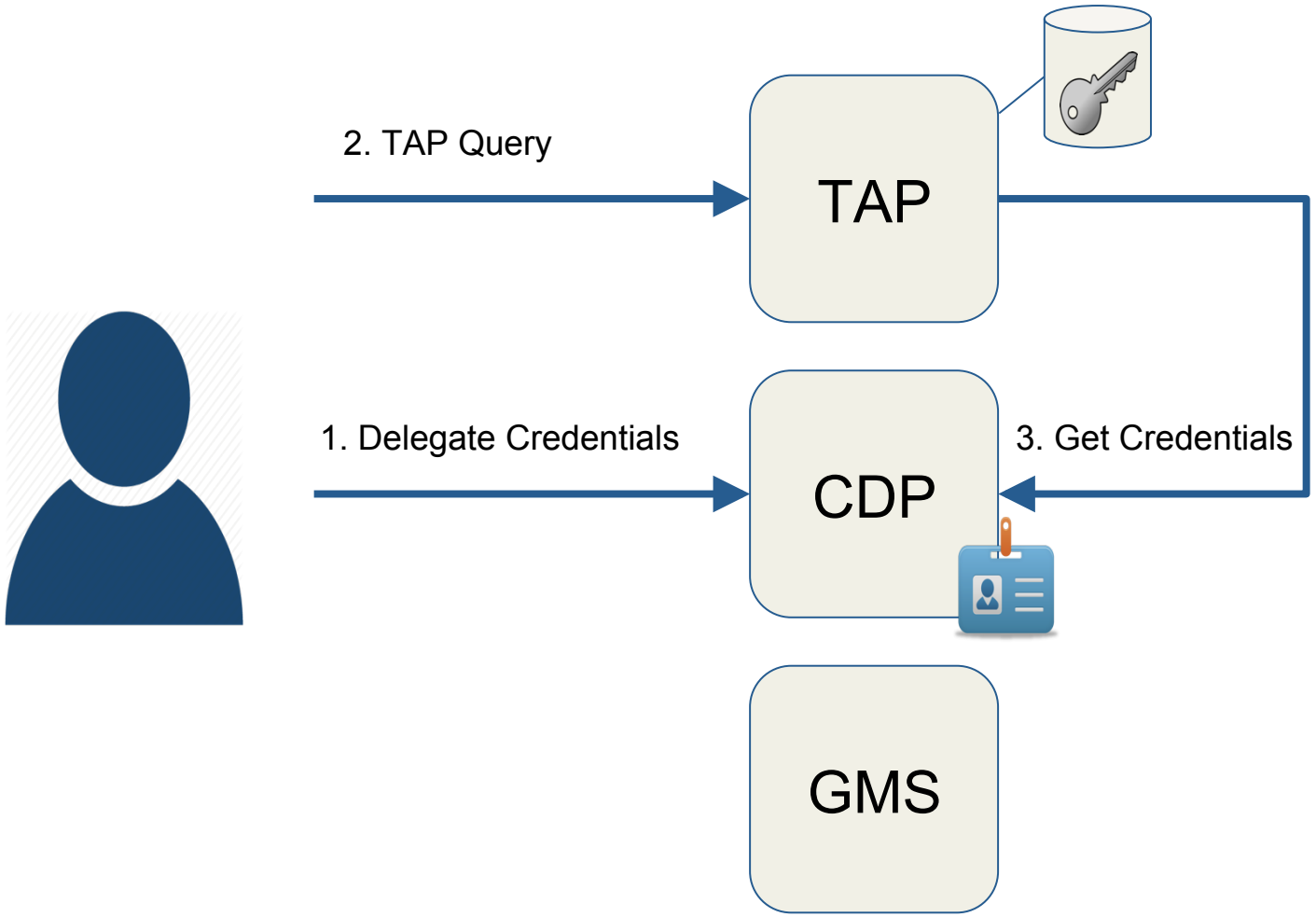
Scenario 1: Authorized TAP Query



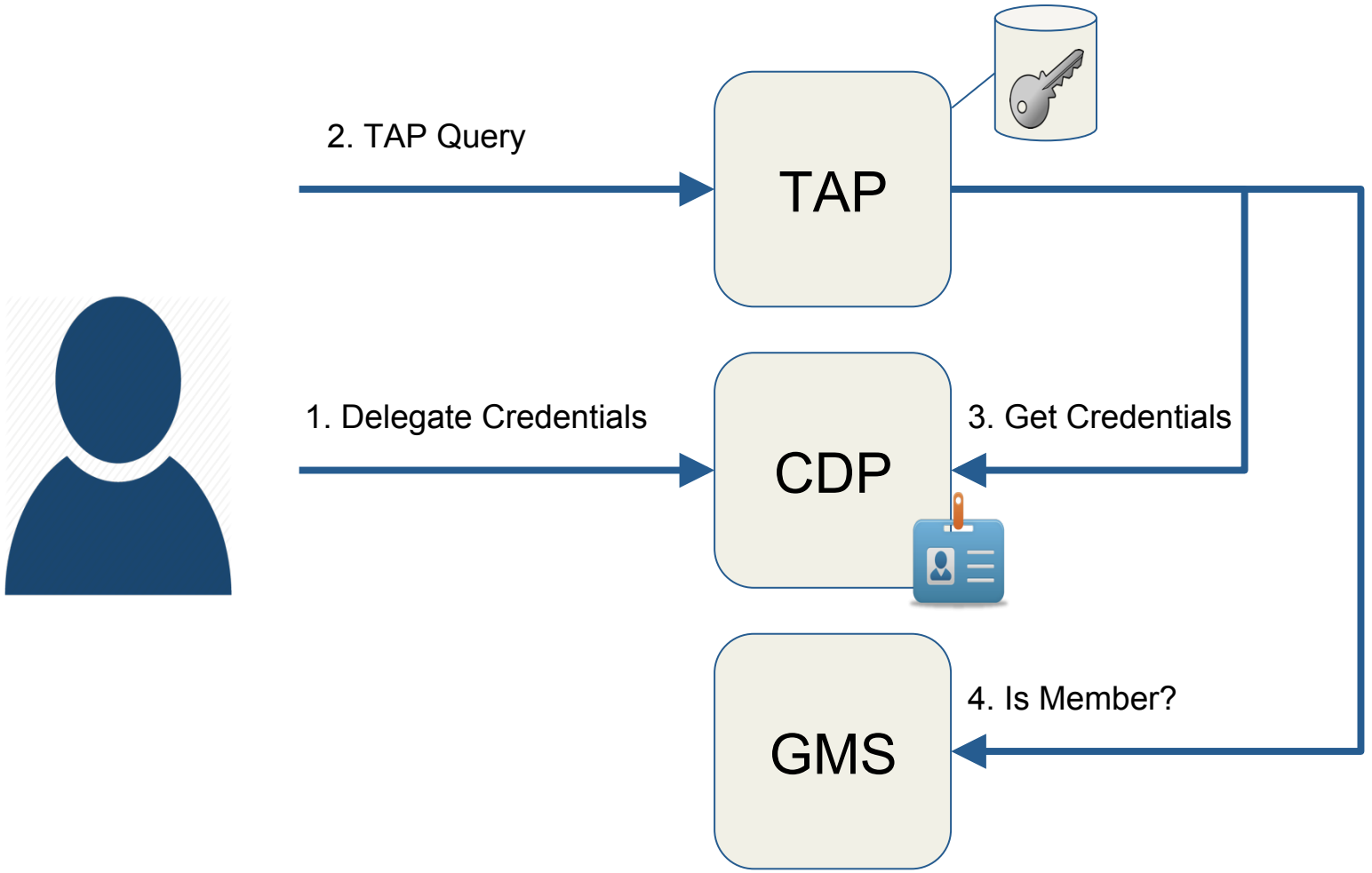
Scenario 1: Authorized TAP Query



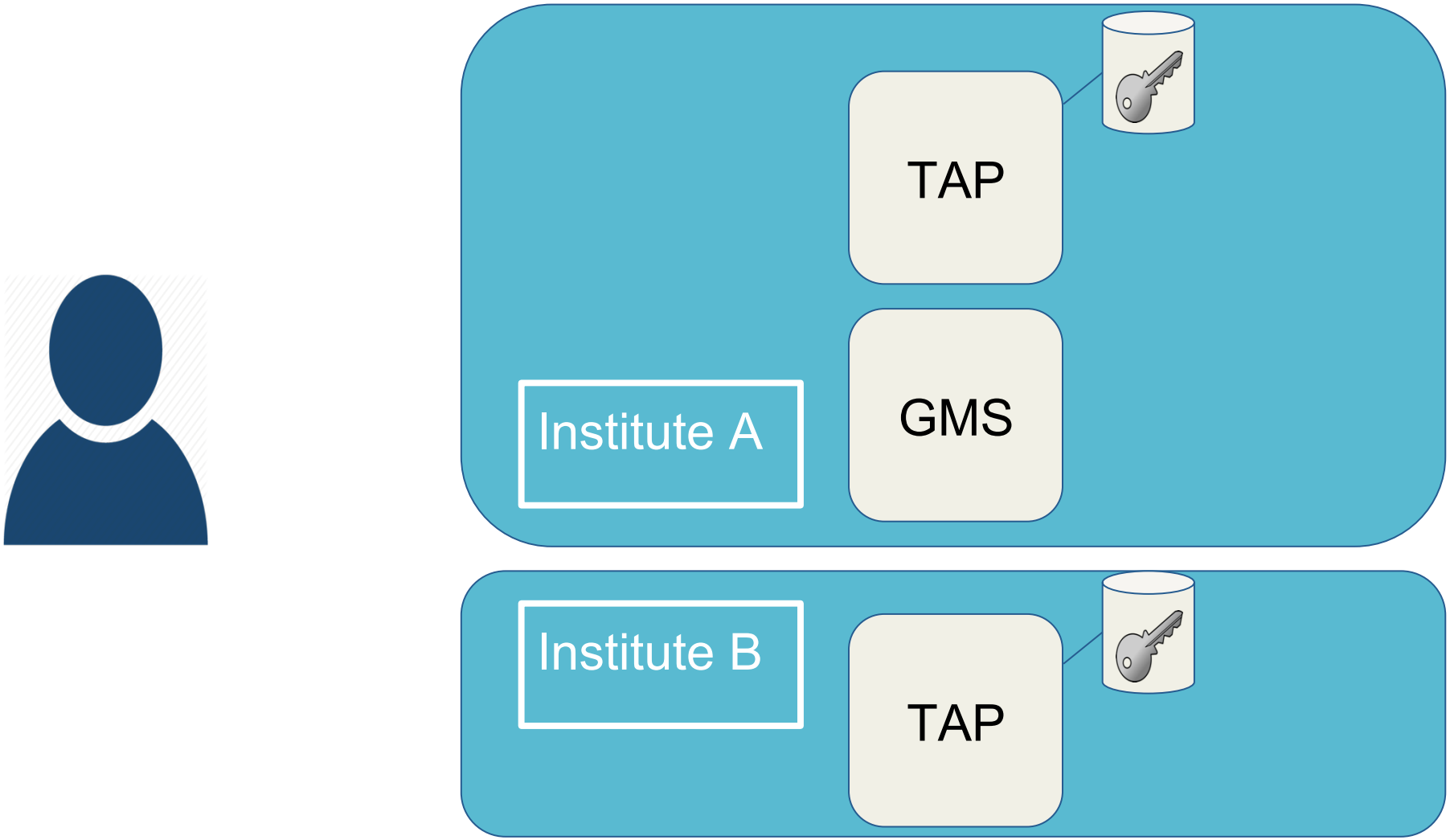
Scenario 1: Authorized TAP Query



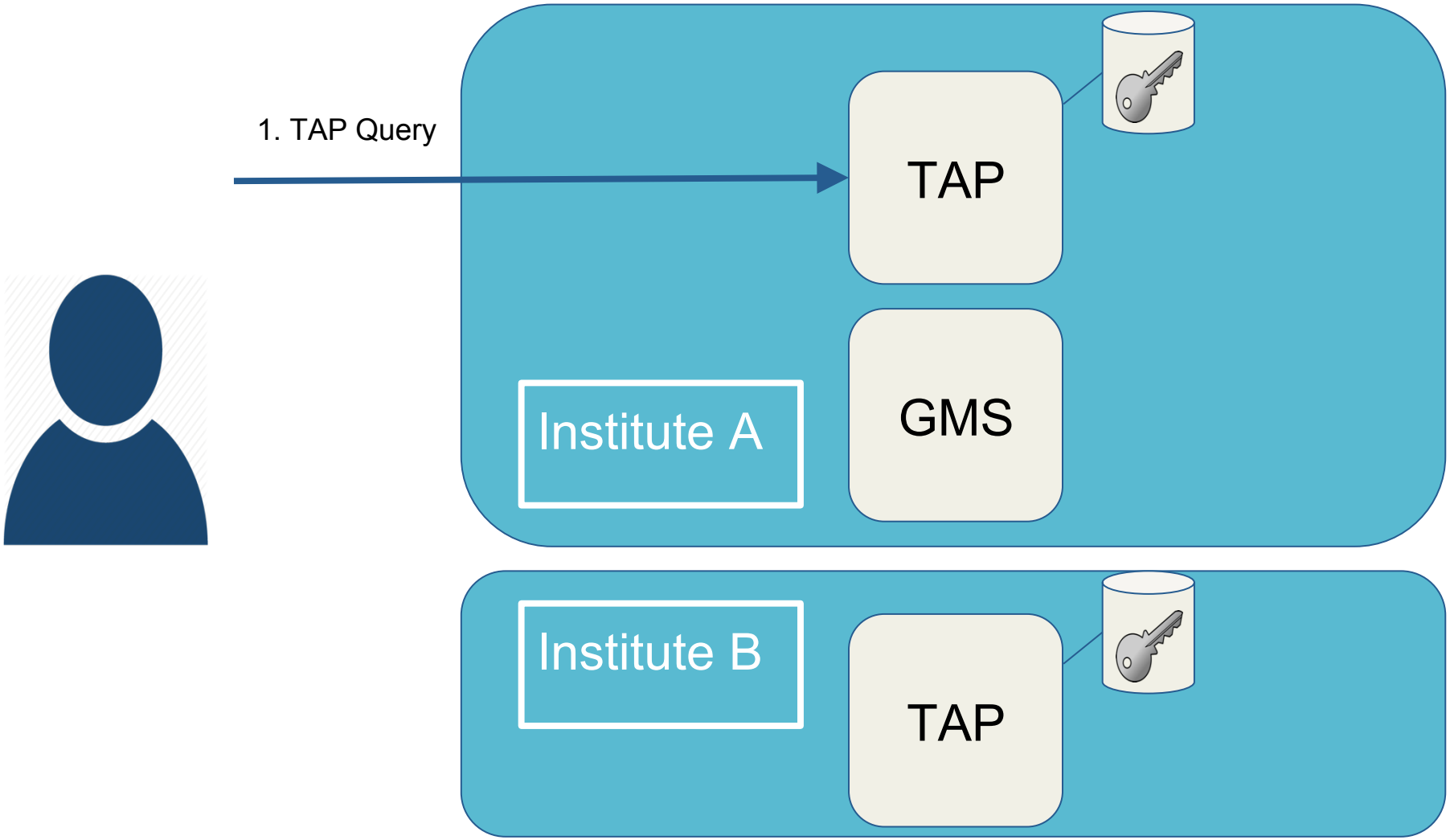
Scenario 1: Authorized TAP Query



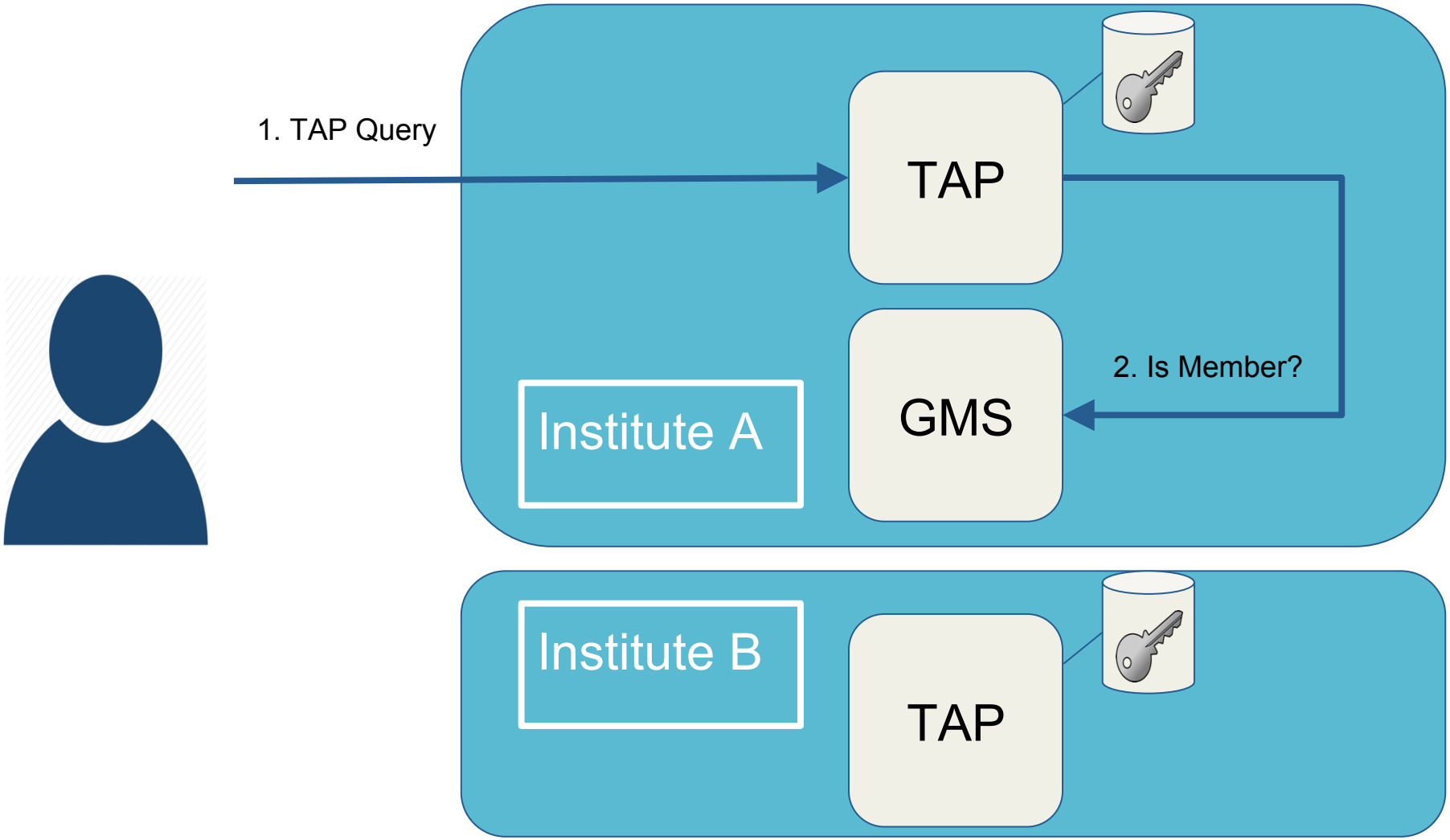
Scenario 2: Distributed TAP Query



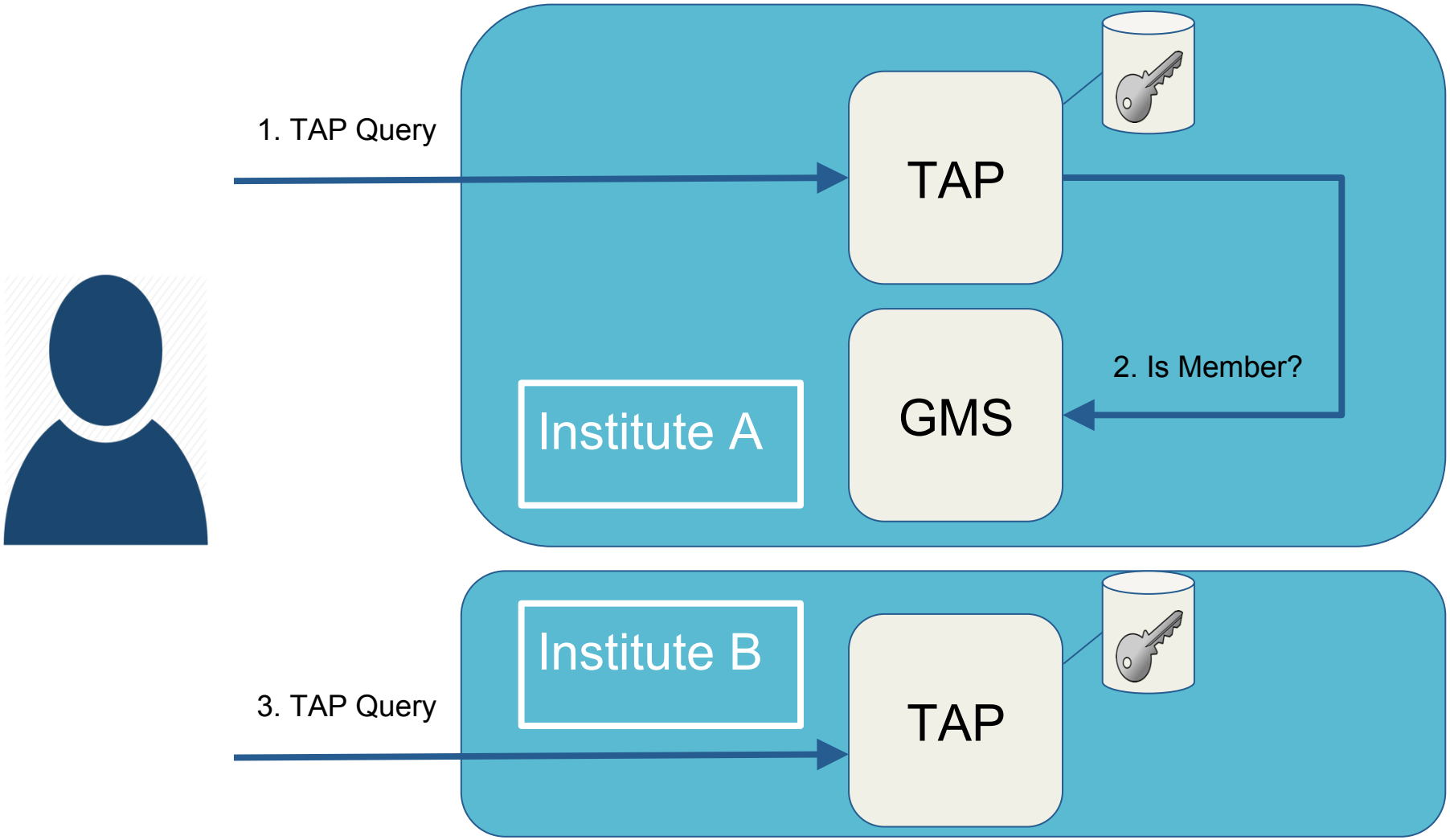
Scenario 2: Distributed TAP Query



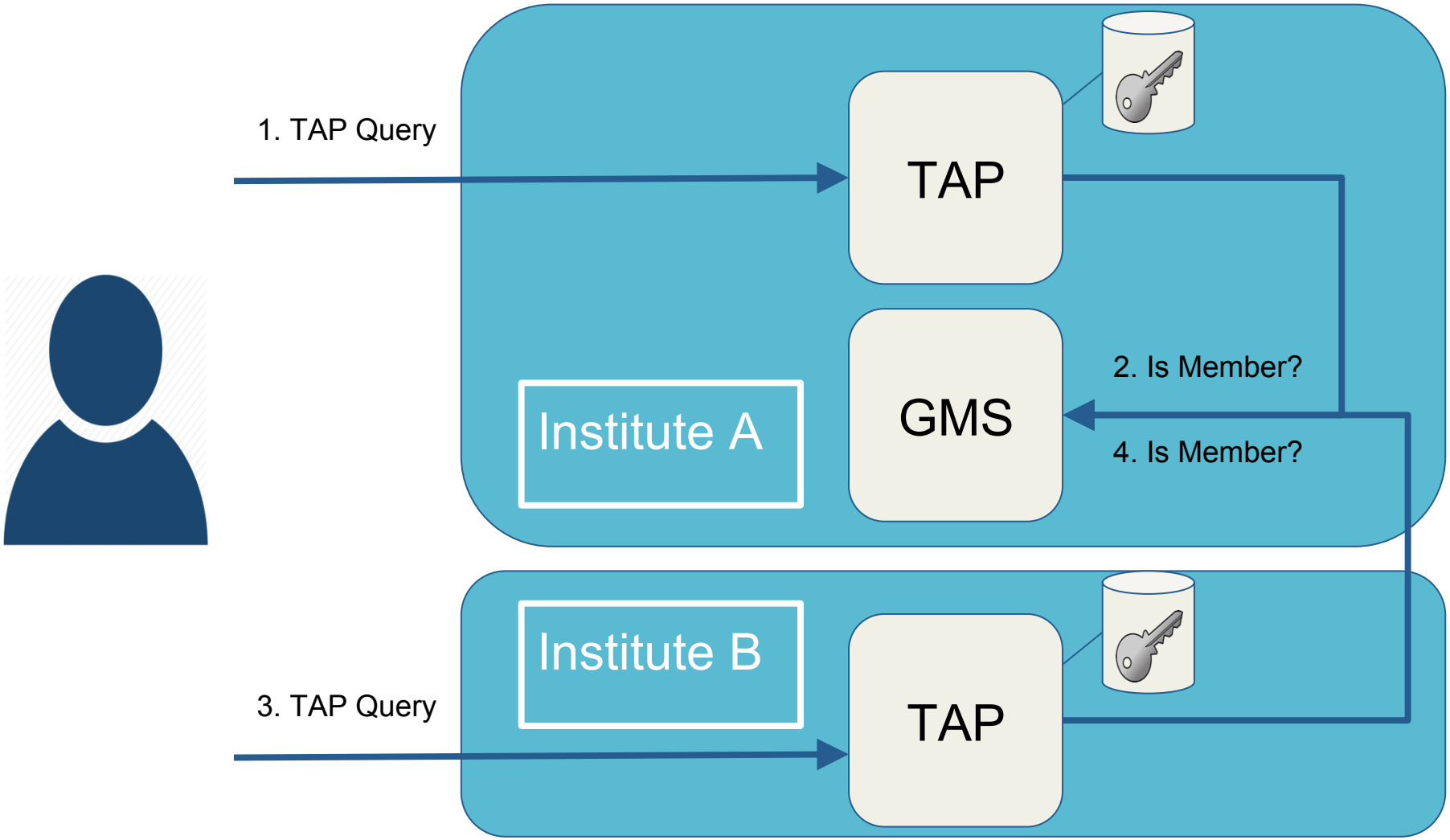
Scenario 2: Distributed TAP Query



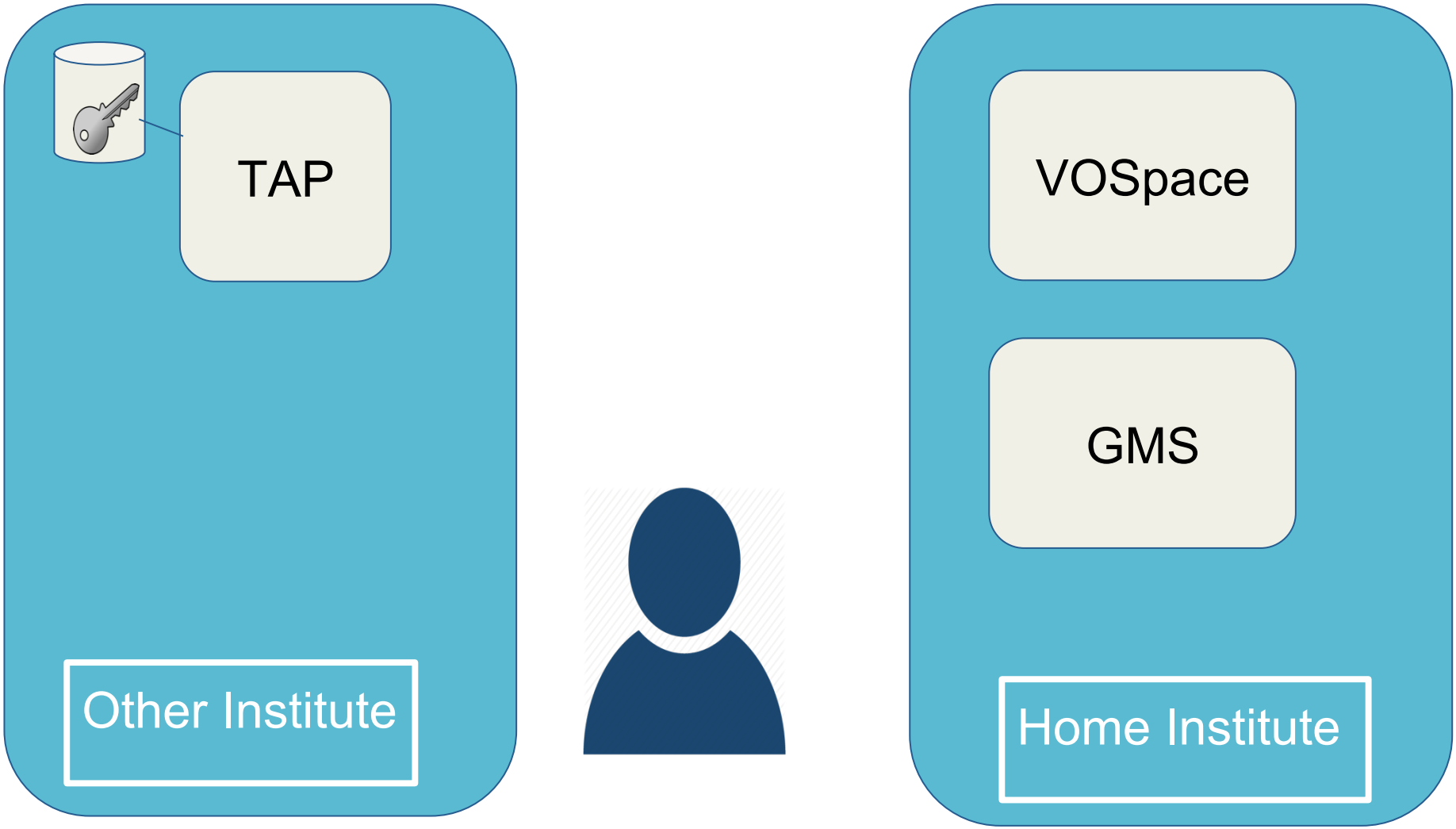
Scenario 2: Distributed TAP Query



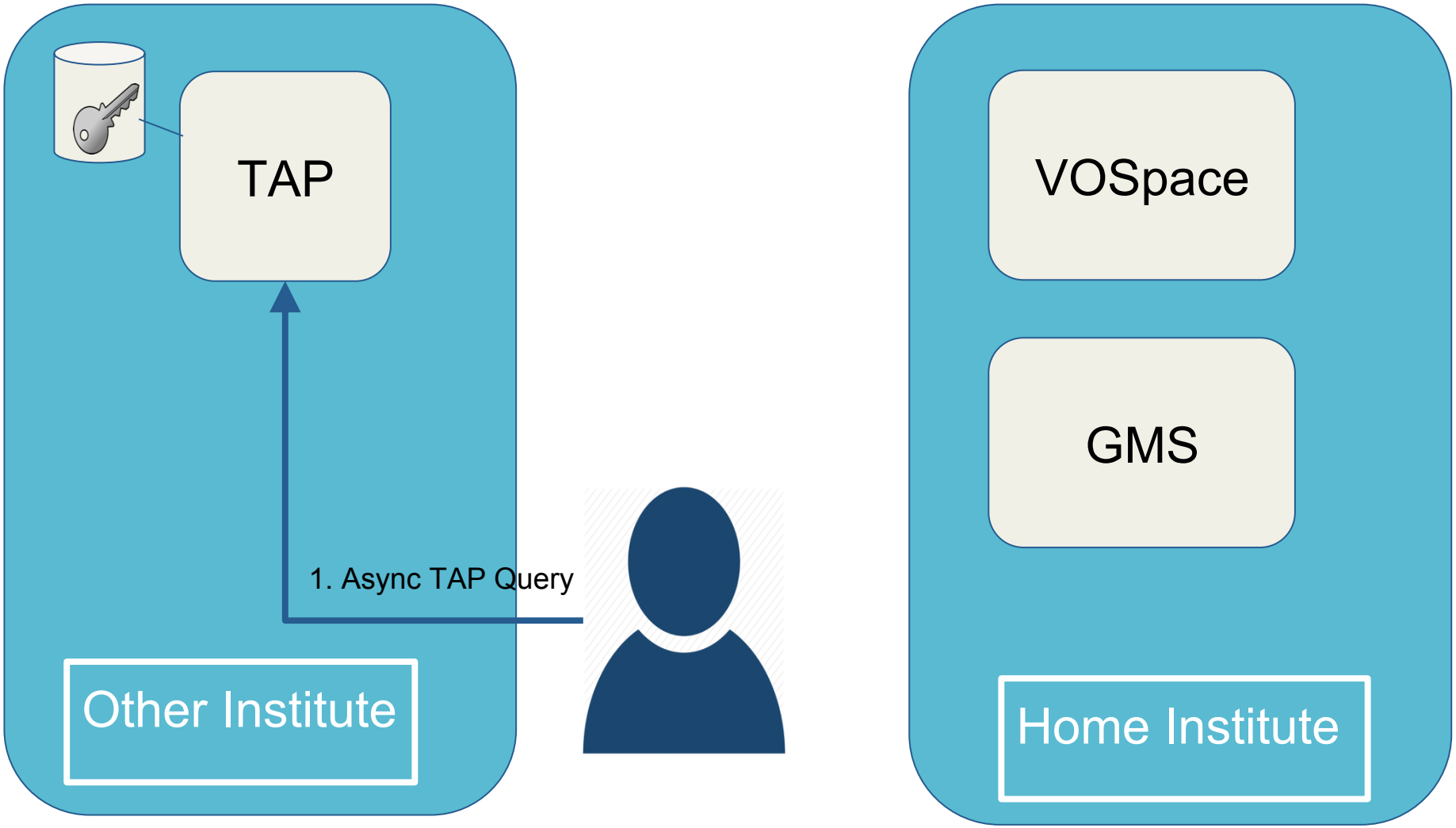
Scenario 2: Distributed TAP Query



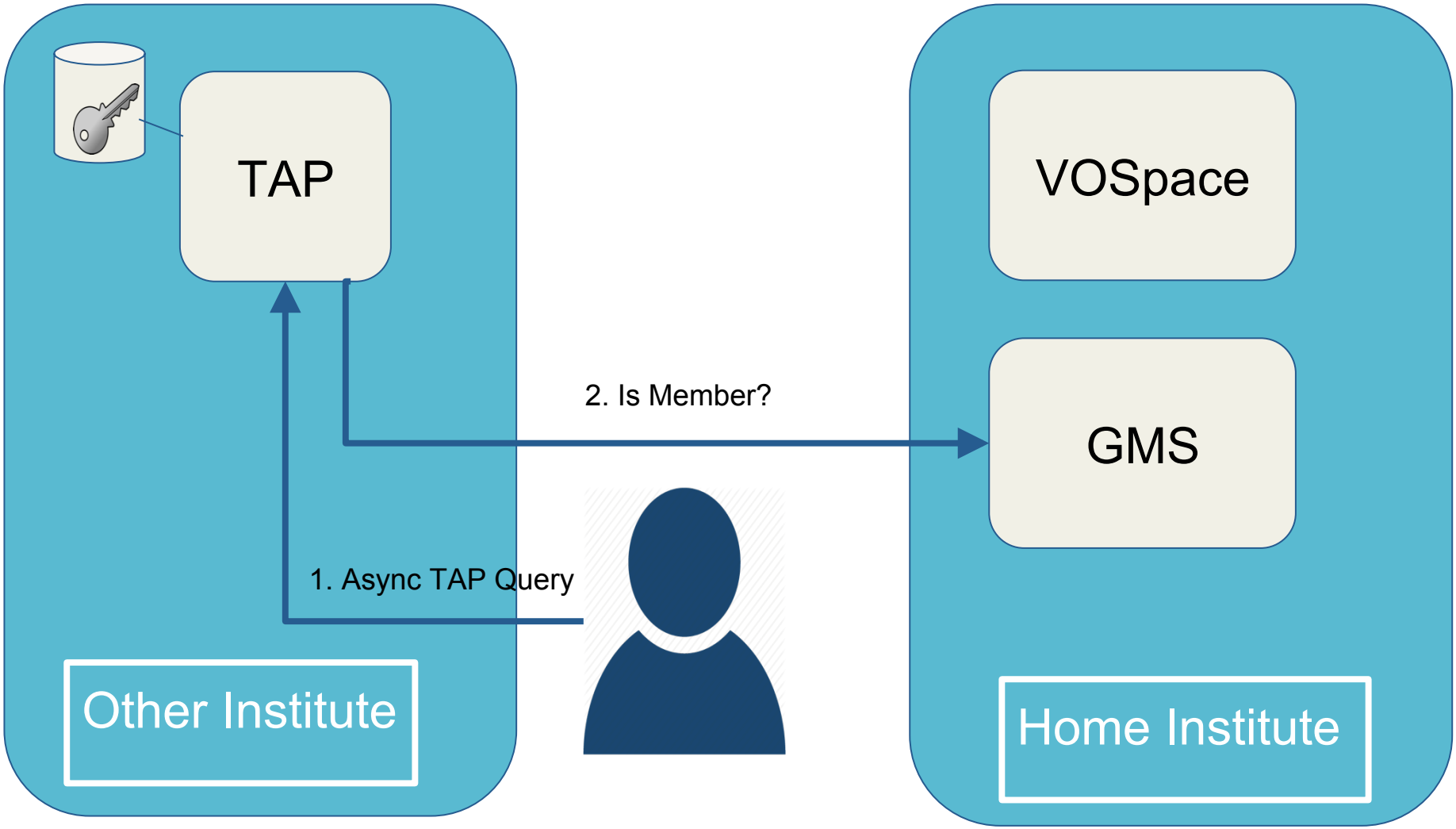
Scenario 3: Remote TAP Result Storage



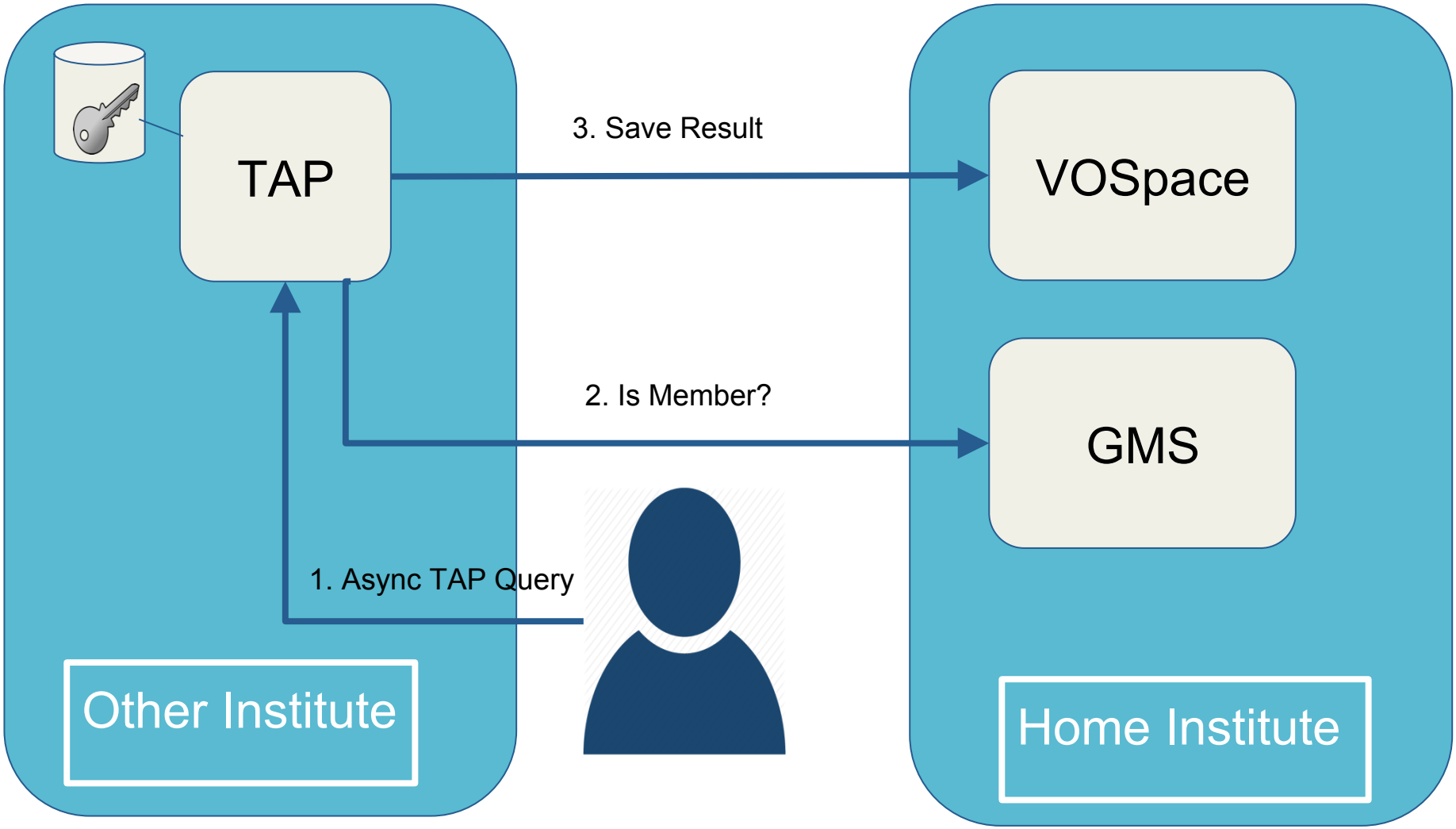
Scenario 3: Remote TAP Result Storage



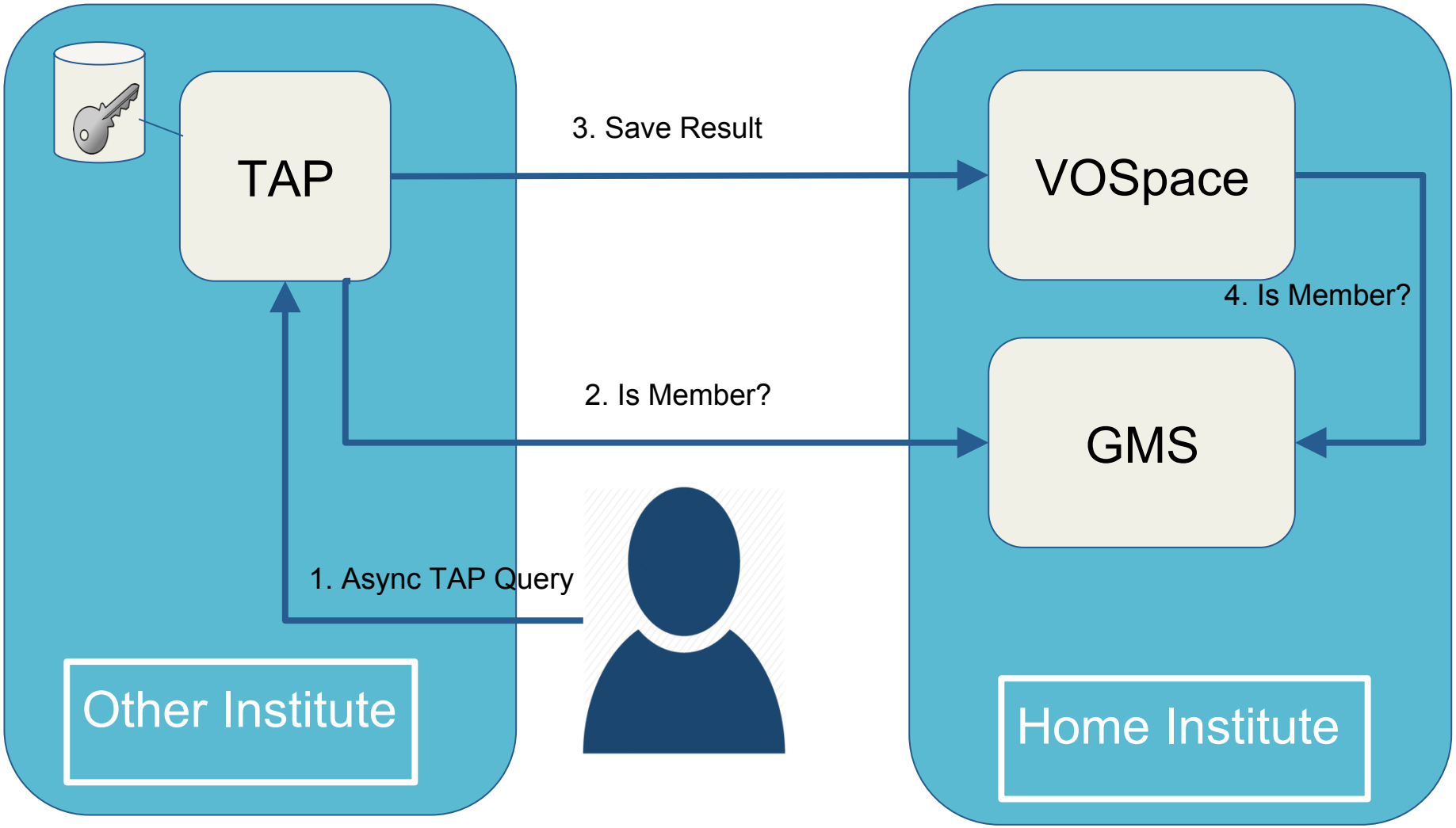
Scenario 3: Remote TAP Result Storage



Scenario 3: Remote TAP Result Storage



Scenario 3: Remote TAP Result Storage



API Definition

Service Operations

Required

- boolean isMember(Group)
- list<Group> getMemberships()

Group Membership Check API Options

```
GET /gms/groups/{group}

GET /gms/groups/{group}/{userID}

GET /gms/groups/{group}/{userID}?idType={idType}

GET /gms/memberships/{userID}

GET /gms/memberships/{userID}?idType={idType}

GET /gms/search/{userID}?role={role}&group={group}
```

Other Notes

GMS Implementation Possibilities

- Via Grouper (groups in MySQL, users in LDAP)
- LDAP only with memberOf plugin (supports groups-of-groups)
- VOSpace implementation:
  - ContainerNodes = groups
  - DataNodes = users

Extensions to GMS

- Email distrubution lists?

Group Management API could be provided:

- create/modify/delete groups
- Add/remove members

CANFAR GMS and UMS API

http://www.canfar.phys.uvic.ca/ac/#/

Discussion Items

- Scope of the service (only isMember, or have membership management operations?)
- Need for a ‘superuser’ isMember call (not the user making the call)
- Groups of groups?
- Discovering the right CDP endpoint
- CDP for other authentication methods? (OAuth2 can do this)