Cross-Realm Authentication

INFN Istituto Nazionale di Fisica Nucleare

Sandro.Angius@lnf.infn.it

OpenAFS

Kerberos Cross-Realm Authentication

Introduction

- Allows users of one Kerberos realm to authenticate with services in other realms
- Implemented by sharing a “secret” that realizes a trust relationship between realms
- Using the shared secret key, KDCs can check the authenticity of cross-realm requests coming from users of trusted realms and then can securely issue service tickets for them

Kerberos Cross-Realm Authentication

Types

- Mono or Bi-directional
- Direct Trust
- Transitive Trust
  - Reduce number of keys to be used
    - $O(n^2)$ instead of $O(2^n)$
- Authentication paths
  - Hierarchical (Default)
  - “CAPATHS” (Need configuration)

Kerberos Cross-Realm Authentication

How?

Cross-Realm Authentication Flow Steps

- When requesting authentication to a service in a foreign realm (i.e.: “service/host.other.org@OTHER.ORG” from “user@MY.ORG”)

1. Client requests a Cross-Realm ticket to local KDC (and, if needed, all Cross-Realm tickets from other KDCs necessary to reach the foreign realm)
2. Client requests a ticket for the requested service to the foreign KDC using the Cross-Realm ticket
3. Client presents the obtained service ticket to the foreign AP server

Kerberos Cross-Realm Authentication

Setup

Kerberos Cross-Realm Authentication

Setup

- To permit Cross-Realm authentication from *REALM1* to *REALM2* just add the following principal on both KDC:
  - krbtgt/*REALM2*@*REALM1*
- The key of the principals must be the same
  - MIT recommends a minimum of 26 chars of random ASCII text
  - If “*Decrypt integrity check failed*” error occurs check them
- The key version number (kvno) must be the same
- Supported encryption types must be compatible
- “*No matching key in entry*” error if kvno or enctypes mismatch

Kerberos Cross-Realm Authentication

Setup

- In the previous example reverse Cross-Realm authentication (*REALM2* to *REALM1)* can be achieved just adding the following principal on both KDC:
  - krbtgt/*REALM1@REALM2*
  - The key of these principals can differ from the one used for the krbtgt/*REALM2@REALM1* principals

Kerberos Cross-Realm Authentication

Setup

- Non-hierarchical transitive Cross-Realm authentication requires to setup [capaths] section in krb5.conf on each host involved allowing clients to find out the path to follow and allowing the server to validate the followed path
  - Error messages related to a [capaths] mis-configuration:
    - “Illegal cross-realm ticket”, “bad realm transit path from…”
  - Use “kvno principal@FOREIGN.REALM” command for a fast check of the Cross-Realm authentication

AFS in a Cross-Realm Environment

- Cross-Realm authentication permits to give (authenticated) access to the AFS service for users belonging to trusted realms
- After completed the Kerberos Cross-Realm authentication, create the Cross-Realm PTS group:

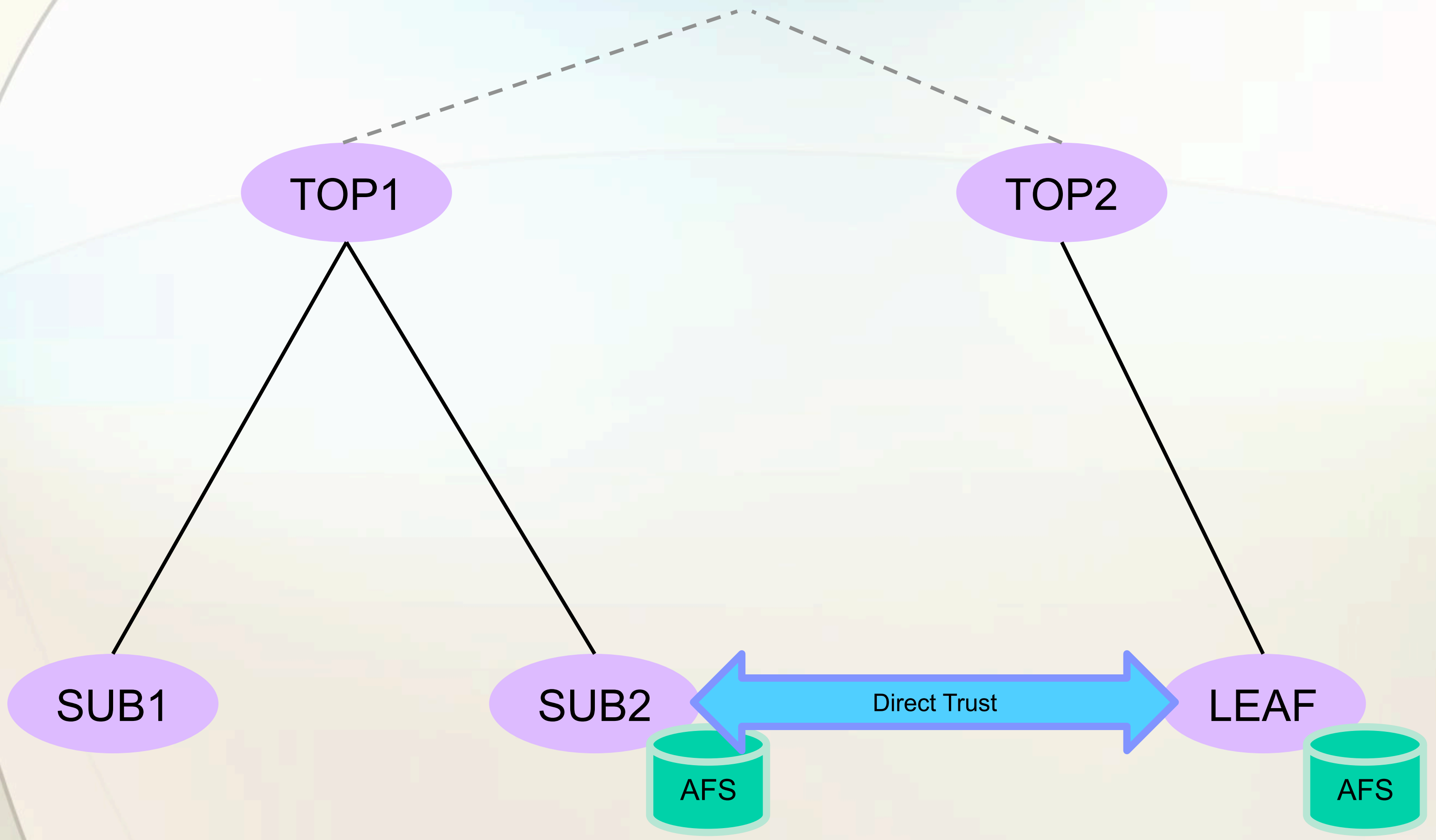
  -
    ```
    pts cg system:authuser@realm1.org
                   -owner system:administrators –cell realm2.org
    ```
- This special group permits to "aklog" command to create the PTS Cross-Realm user

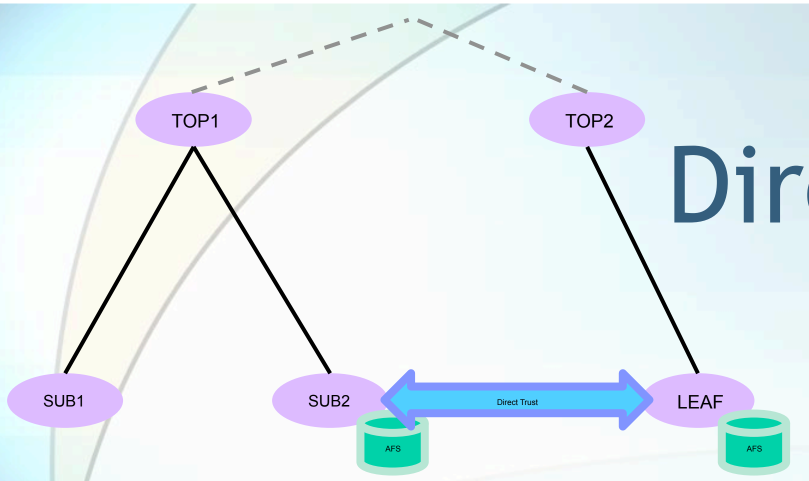
AFS Cross-Realm Users

- Have a high PTS id
- Name is like:
  - username@realm1.org
- Cannot be added to Bos UserList
- Not members of system:authuser
- Members of system:authsuser@realm1.org

Examples

Direct Trust





Direct Trust

On both LEAF and SUB2 realms create the Cross-Realm principals:

```
kadmin -q "addprinc krbtgt/SUB2.TOP1@LEAF.TOP2”
kadmin -q "addprinc krbtgt/LEAF.TOP2@SUB2.TOP1”
```

Check trust using kvno, for example with a SUB2 credential against a LEAF user:

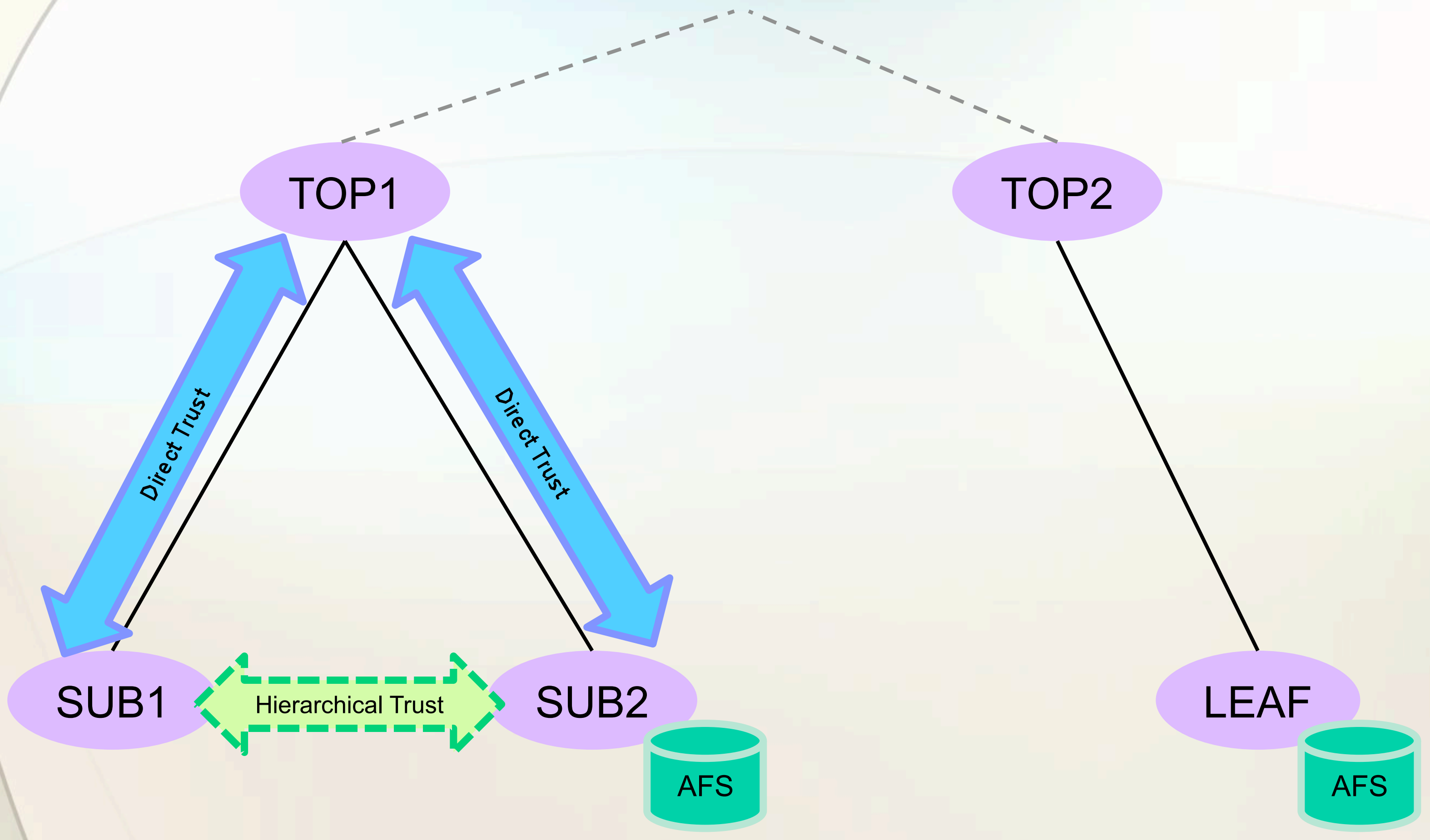
```
kvno leafuser@LEAF.TOP2
```

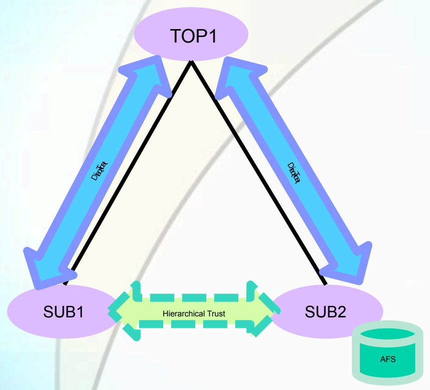
If it works, with a klist you should have something like:

```
Default principal: sub2user@SUB2.TOP1

Valid starting     Expires            Service principal
09/16/09 17:03:53  09/17/09 17:03:53  krbtgt/SUB2.TOP1@SUB2.TOP1 # sub2user ticket
09/16/09 17:04:08  09/17/09 17:03:53  krbtgt/LEAF.TOP2@SUB2.TOP1 # Cross-Realm ticket
09/16/09 17:04:08  09/17/09 17:03:53  leafuser@LEAF.TOP2         # Foreign ticket
```

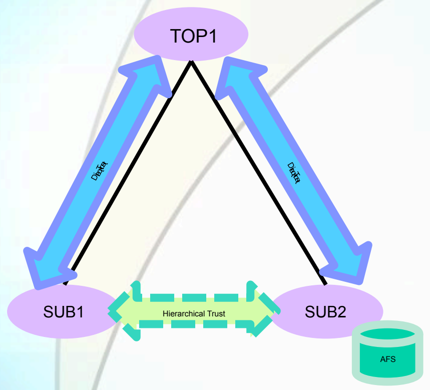
Hierarchical Trust





Hierarchical Trust

- Using the convention of naming realms with the name of DNS domains in upper case permits (with the transitivity property implemented in Kerberos5) to extend Cross-Realm authentication to an entire tree just defining direct trusts between adjacent realms (domains).
- In this example defining a direct trust between SUB1 and TOP1 and another direct trust between SUB2 and TOP1, will realize also a trust between SUB1 and SUB2 realms.
- Note: adding a new realm in the tree hierarchy (i.e. SUB3.TOP1) and defining a direct trust between this new realm and its parent realm (TOP1) will result in an implicit trust between the new realm (SUB3) and the older ones (SUB1 and SUB2 in this example)



Hierarchical Trust

On both SUB1 and TOP1 realms create the Cross-Realm principals:

```
kadmin -q "addprinc krbtgt/SUB1.TOP1@TOP1”
kadmin -q "addprinc krbtgt/TOP1@SUB1.TOP1”
```

On both SUB2 and TOP1 realms create the Cross-Realm principals:

```
kadmin -q "addprinc krbtgt/SUB2.TOP1@TOP1”
kadmin -q "addprinc krbtgt/TOP1@SUB2.TOP1”
```

Check trust using kvno, for example with a SUB1 credential against a SUB2 user:

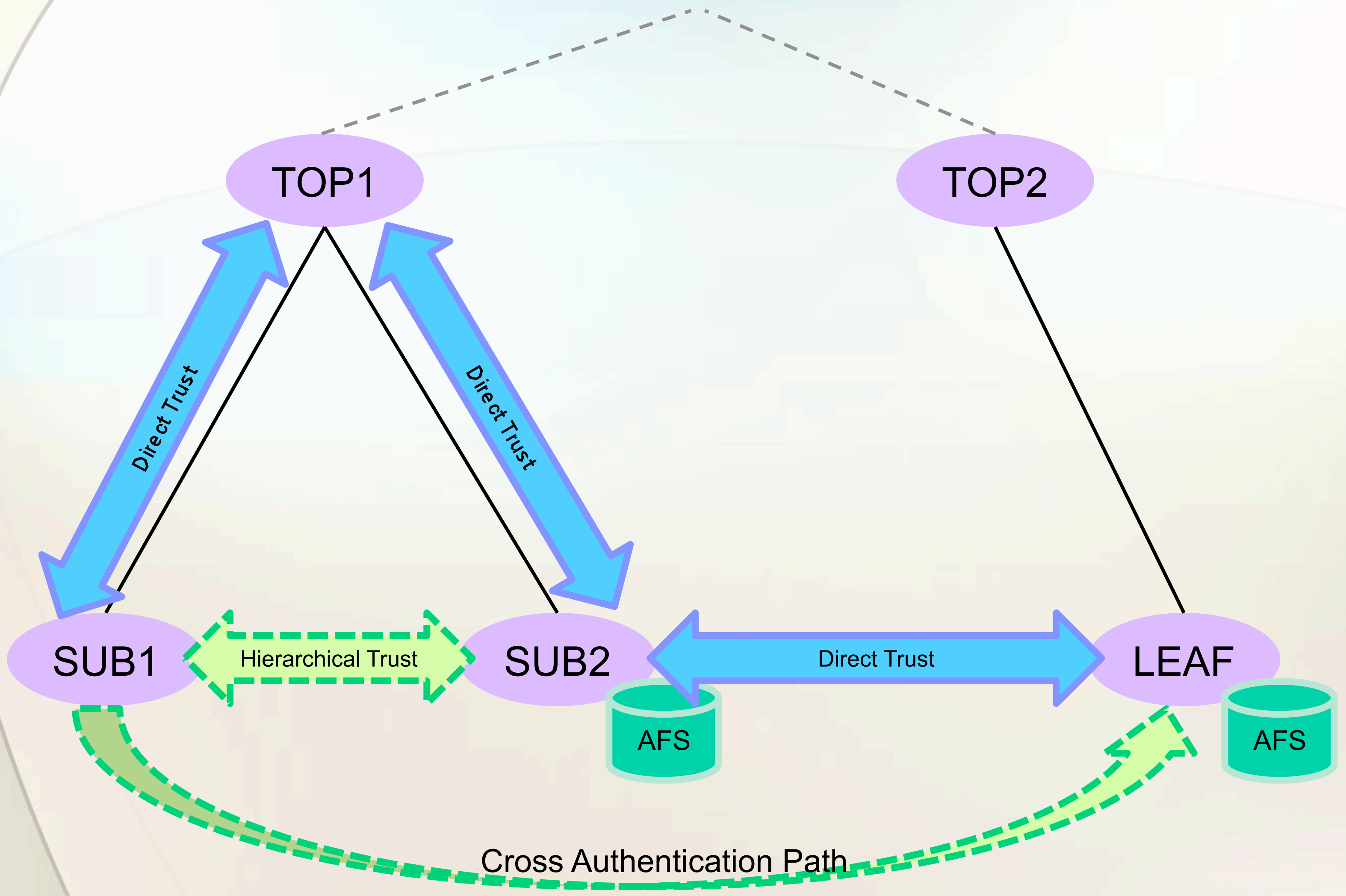
```
kvno sub2user@SUB2.TOP1
```

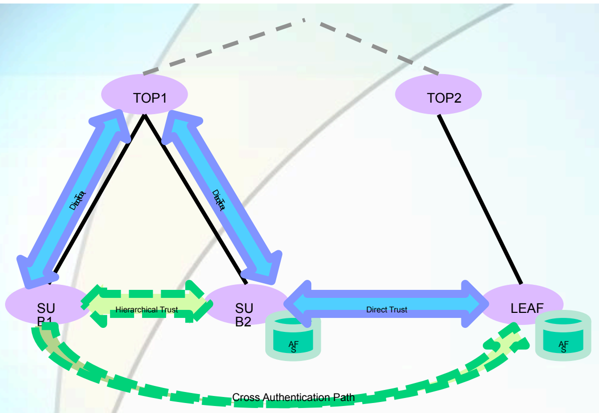
If it works, with a klist you should have something like:

```
Default principal: sub1user@SUB1.TOP1

Valid starting     Expires            Service principal
09/23/09 17:46:33  09/24/09 17:46:33  krbtgt/SUB1.TOP1@SUB1.TOP1     # sub1user ticket
09/23/09 17:46:59  09/24/09 17:46:33  krbtgt/TOP1@SUB1.TOP1          # 1st Cross-Realm Ticket
09/23/09 17:46:59  09/24/09 17:46:33  krbtgt/SUB2.TOP1@TOP1          # 2nd Cross-Realm Ticket
09/23/09 17:46:59  09/24/09 17:46:33  sub2user@SUB2.TOP1             # Foreign ticket
```

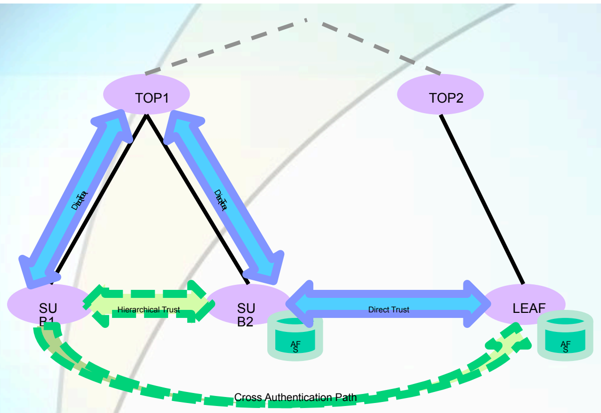
CAPATHS Trust





CAPATHS Trust

- Cross authentication paths (CAPATHS) permit to use transitivity Cross-Realm authentication when hierarchy isn't possible or to speed up authentication phase (limiting the number of realms to traverse)
- Previous examples have defined three direct trust between:
  - SUB1 and TOP1, SUB2 and TOP1, SUB2 and LEAF
- Trust between SUB1.TOP1 and LEAF.TOP2 can be defined just adding few lines in [capaths] section of the krb5.conf file



CAPATHS Trust

```
[capaths]
 SUB1.TOP1 = {                        # For client of SUB1.TOP1 realm
       LEAF.TOP2 = TOP1               # 1st step to reach LEAF.TOP2 realm
       LEAF.TOP2 = SUB2.TOP1          # 2nd step to reach LEAF.TOP2 realm
 }
```

After a “kvno leafuser@LEAF.TOP2”, klist will show something like:

```
Default principal: sub1user@SUB1.TOP1

Valid starting     Expires            Service principal
09/24/09 18:11:31  09/25/09 18:11:31  krbtgt/SUB1.TOP1@SUB1.TOP1    # sub1user ticket
09/24/09 18:11:40  09/25/09 18:11:31  krbtgt/TOP1@SUB1.TOP1         # 1st Cross-Realm Ticket
09/24/09 18:11:40  09/25/09 18:11:31  krbtgt/SUB2.TOP1@TOP1         # 2nd Cross-Realm Ticket
09/24/09 18:11:40  09/25/09 18:11:31  krbtgt/LEAF.TOP2@SUB2.TOP1    # 3rd Cross-Realm Ticket
09/24/09 18:11:40  09/25/09 18:11:31  leafuser@LEAF.TOP2            # Foreign ticket
```

Using AFS in Cross-Realm Environment

Add group system:authuser@sub1.top1 on leaf.top2 AFS cell:

pts cg system:authuser@sub1.top1 -o system:administrators –c leaf.top2

Adding “-d” to aklog for some debug information you shoud see something like:

```
aklog -d -cell leaf.top2

Authenticating to cell leaf.top2 (server afs1.leaf.top2).
Trying to authenticate to user's realm SUB1.TOP1.
Getting tickets: afs/leaf.top2@SUB1.TOP1
We've deduced that we need to authenticate to realm LEAF.TOP2.
Getting tickets: afs/leaf.top2@LEAF.TOP2
Getting tickets: afs/leaf.top2@LEAF.TOP2
Getting tickets: afs@LEAF.TOP2
Using Kerberos V5 ticket natively
About to resolve name sub1user@SUB1.TOP1 to id in cell leaf.top2.
Id 32766
doing first-time registration of sub1user@sub1.top1 at leaf.top2
created cross-cell entry for sub1user@sub1.top1 (Id 130866) at leaf.top2
Set username to AFS ID 130866
Setting tokens. AFS ID 130866 /  @ SUB1.TOP1
```

Using AFS in Cross-Realm Environment

The “tokens” command will show:

```
Tokens held by the Cache Manager:

User's (AFS ID 130866) tokens for afs@leaf.top2 [Expires Sep 18 18:09]
   --End of list--
```

The Cross-Realm user has been added to the Cross-Realm pts special group:

```
pts mem system:authuser@sub1.top1 -cell leaf.top2

Members of system:authuser@sub1.top1 (id: -206) are:
  sub1user@sub1.top1
```

Group system:authuser@sub1.top1 can be used in ACLs and Cross-Realm usernames like sub1user@sub1.top1 can be used in ACLs or added to groups

When Cross-Realm is not involved…

- Previous examples regard environment with multiple kerberos realms who want to establish a trust relationship to share services; with AFS two other kinds of environments are possible:
    - Multiple AFS Cells in one Kerberos Realm
    - Multiple Kerberos Realms with one AFS Cell

Multiple AFS Cells in one K5 Realm

- Because no relationship is needed between AFS cell name and Kerberos realm name, one centralized Kerberos realm can be used to serve authentication to multiple departmental AFS cells
- To deploy this environment
  - Put the centralized Kerberos realm name on the first line of /usr/afs/etc/krb.conf file on each AFS cell
  - For each AFS cell name add a service principal named "afs/cellname@REALM" in the centralized Kerberos realm

Multiple K5 Realms with one AFS Cell

- Centralized naming and account allocation management with multiple Kerberos realm synchronized to reflect account allocation
- Any of the “equivalent” principal want to access AFS cell with the same privileges regardless the realm that issued authentication ticket
- To deploy this environment
  - Put all Kerberos realm names on the first line of /usr/afs/etc/krb.conf file of the AFS cell

Useful Resources

- MIT Kerberos V5 System Administrator's Guide
  - http://web.mit.edu/Kerberos/krb5-1.7/krb5-1.7/doc/krb5-admin.html
- Frequently Asked Questions about the Kerberos
  - http://www.cmf.nrl.navy.mil/ccs/people/kenh/kerberos-faq.html
- Kerberos Protocol Tutorial
  - http://www.kerberos.org/software/tutorial.html
- OpenAFS BPW09 Kerberos5 Tutorial

Thank you

Sandro.Angius@lnf.infn.it