Chapter 4

EXPERIMENTAL SETUP
Chapter 4: Experimental Setup

4. Experimental Setup

4.1 Hardware and Software Setup

4.1.1 Installing Server Components

4.1.2 Installing Client Components

4.1.3 Data Design

4.1.4 Server Side Data design

4.1.5 Storage

4.1.6 User Side Data design

4.1.7 Data Model and DB Design

4.1.8 Kerberos Server Data Storage

4.2 System Implementation Components

4.2.1 KDC Server - Implementation Setup

4.2.2 Kerberos Client – Setup
Chapter 4

EXPERIMENTAL SETUP

4.1 Hardware and software setup

The proposed system consists of two servers that is the KDC server and the Apache server. We have added an additional component called as the LDAP Agent and it will run on Apache DS [18]. On the same server a location validator component is also added. The reason to have these components together is, in-memory reference and to avoid I/O latency. These components are written purely in java.

The next component is the KDC component i.e AS and TGS. Their behaviors are modified through a KDC config file so that these components automatically contact LDAP. Next is the Tomcat HTTP server component called TestAppServlet. This is also linked to LDAP for dynamic authorization. A Keytab file is used here to validate an AP request. As stated earlier, the system can be seen as a server component and a client component.

4.1.1 Installing Server Components

KDC Server, App Server, Location Service, Apache DS LDAP Database on one server.
• All servers/services are deployed at the server where the Fedora 16 Operating System is running. This server is to be directly connected to the Internet through Broadband connection.

• **KDC Server:** Provides standard commands to start and monitor KDC tickets, keys, and KDC configuration files (KDC database).

• **Fedora 16 OS:** It is Linux based OS and can be operated by any person who understands how to use UNIX/LINUX based OS

• **Apache DS LDAP DB** – ApacheDS is an extensible and embeddable directory server entirely written in Java.

• **Apache Tomcat Web Server** – It’s a simple web server designed for running Java Web based applications. This web-server can be started or stopped using a browser. It runs on the HTTP protocol.

### 4.1.2 Installing Client Components (Personal Laptop)

• All client utilities like Kerberos Client, Smart Card Reader Interface, Bluetooth, Wireless Internet Card Drivers, and related software’s are installed on personal Laptop.

• Device is having Android OS, Bluetooth connectivity, and open source API to retrieve GPS location.

ApacheDS 2.0 is used for Server side modifications. ApacheDS is an extensible directory server entirely written in the Java programming language. It has been certified by the Open Group and is compatible with LDAPv3. ApacheDS supports Standard Kerberos V, LDAP, and the Change Password
Prototype implementation is done as shown in Figure 4.1, created in a local environment.

Fig. 4.1: Experimental setup of the PKLK System.

It consists of the Apache Directory Server along with Local Client, GPS Device, and the Test Application server. The database is implemented using LDAP. The Apache Directory Server is used and it has an inbuilt support for KDC. The Apache Directory Server uses the LDAP protocol. We store user information, device identification, and test application server information. Location Manager is written in the java programming language and it is extended with the KDC Server for testing. We have developed a standalone Kerberos client. The client consists of two major components i.e.,
authentication and authorization. The GPS device simulator component is written in Java. The Users setup is done using a LDIF file. It includes all the attributes of a user. It gets imported using the Apache Directory Utility in the LDAP database.

A private and public key is generated using the Command line JDK based key tool. The Keystore.jks file acts as a repository for storing security Certificates. Location Device setup is done using the custom context device. A private key is securely burned into the context device hardware. Each device is assigned a unique key and is registered in LDAP against the respective user. We have written policies in a policy file. The policy file stores allowed regions, and location co-ordinates.

### 4.1.3 Data Design

For this Research, the PKLK system’s data design is divided in to two parts: Server side and Client side. The Server Side data design consists of Users data, Services Data, and Context/Policy Data. Ideally, this can be stored separately, but for Research we store it in a simple text file. On the User Side a data design consists of Personal Data Storage that is stored onto the User’s machine and/or Smart Card.

### 4.1.4 Server Side Data design

PKLK server components use the LDAP database provided by the Apache Directory Server. We store everything related to users, hosts, and services in
the LDAP base, as entries. In order to be able to retrieve them, we have to store them in a known place in the hierarchy. There is an existing LDAP schema to manage the keys and other information, named krb5kdc. It contains 3 Object Classes and 15 Attribute Types. All the Object Classes are auxiliary.

4.1.5 Storage

We store everything related to users, hosts, and services in the LDAP database as entries. In order to be able to retrieve them, we have to store them in a known place in the hierarchy. This position is known by the Kerberos server using the Search Base DN parameter.

Every time the Kerberos server receives a request for a ticket from a principal, it will do a LDAP search starting from the Search Base DN, looking for any entry matching the filter ‘(krb5PrincipalName=)’. This entry should contain the Kerberos keys that will be used to generate the ticket.

One more requirement is that the key should contain the version number which will allow a user to keep going with a previous key when he just changed its password (an operation that will change the Kerberos keys).

So, for an LDAP entry to be seen as a valid Kerberos entry, it has to contain a Krb5PrincipalName, a Krb5Key, and one more attribute, the Krb5KeyVersionNumber. Table 4.1 shows attribute types and its description.

**Table 4.1: LDAP Attributes**

<table>
<thead>
<tr>
<th>Attribute Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>krb5ValidStart</td>
<td>The date at, which the keys are</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>krb5ValidEnd</td>
<td>The date at which the keys aren't valid any more.</td>
</tr>
<tr>
<td>krb5PasswordEnd</td>
<td>The end of password validity.</td>
</tr>
<tr>
<td>krb5MaxLife</td>
<td>The maximum duration.</td>
</tr>
<tr>
<td>krb5MaxRenew</td>
<td>The maximum number of renews.</td>
</tr>
<tr>
<td>krb5KDCFlags</td>
<td>The KDC flags.</td>
</tr>
<tr>
<td>krb5EncryptionType</td>
<td>The EncryptionTypes.</td>
</tr>
<tr>
<td>krb5Key</td>
<td>The generated keys.</td>
</tr>
<tr>
<td>krb5AccountDisabled</td>
<td>The account has been disabled.</td>
</tr>
<tr>
<td>krb5AccountLockedOut</td>
<td>The account has been locked out.</td>
</tr>
<tr>
<td>krb5AccountExpirationTime</td>
<td>The account expiration time.</td>
</tr>
</tbody>
</table>

**krb5Principal** - This Object Class is used to store a Principal. It contains one mandatory Attribute Type, `krb5PrincipalName`, and two optional (`cn` and `krb5PrincipalRealm`).

**krb5Realm** - This Object Class describes a Kerberos Realm. It just contains the Realm's name (`krb5RealName` AttributeType).

**krb5KDCEntry** - This Object Class is used to store all the information needed to manage a Kerberos user or service. It has one mandatory AttributeType, `krb5KeyVersionNumber`, which is set to 0 for newly created users.
or services, and is incremented after each modification is done on the password (which leads to the generation of new keys). Here, is a list of optional Attribute types the entry can have as shown in Figure 4.1.

4.1.6 User Side Data design:

To use the PKLK system, the user must carry the below items on his machine or on the device like a Smart card.

- Keystore repository – This is where the user will securely keep the Private Key and Public Key in an encrypted format.
- Device data – Ideally, this data will be embedded in the Custom GPS Device, but for Research purposes we use file storage, which consists of a device id, and a device key.

4.1.7 Data Model and DB Design

LDAP DB DESIGN:

ApacheDS is an extensible and embeddable directory server entirely written in Java. Besides LDAP it supports the Kerberos 5 and the Change Password Protocol.
Figure 4:2 gives the idea of how user principles and attributes are stored on the server. For example, if there is an employee and we have to keep his data on a server, then the following operations are executed in LDAP:

- Creation of entity
- Modification of entity
- Deletion of entity
- Searching of entity

Whenever there is need to provide for storing data, LDAP is used. It is optimized for read and search operations.

4.1.8 Kerberos Server Data Storage

KDC Master Database for Application Server Entries consists of entries
for all services that require authentication. This service must be registered in the Kerberos database, using the proper service name.

![Diagram of Kerberos Database]

**Fig. 4.3: Kerberos Database.**

When generated, the ticket contains an encrypted structure (EncTicketPart), generated by the server, which is used to define the ticket characteristics as shown in Figure 4.3.

Example: host/myserver.mit.edu@ATHENA.MIT.EDU

host: type of service being offered (it can be ftp, http)

myserver.mit.edu: machine that is offering this service

ATHENA.MIT.EDU: realm name
Storing Service Key

**KDC**: Service key is stored in the Kerberos database.

**Application Server**: Service keys are stored in key tables, which are files known as **keytabs**. `/etc/krb5.keytab` service key is the equivalent of the service's password, and must be kept secure.

The structure of a Kerberos ticket

The Kerberos Ticket is one of the most fundamental elements. It contains all the information a client sends to a service in order to gain access to this service. It is generated by the KDC when a user requests it.

Figure 4:4 represents the different attributes of the Kerberos ticket.

![Kerberos Ticket Structure](image)

**Fig. 4.4: Kerberos Ticket Structure**
Ticket: This field is a ticket authenticating the client to the server.

A user's long-term key is derived from a password. When the user logs on, for example, the Kerberos client on her workstation accepts her password and then converts it to a cryptographic key by passing the text of the password through a one-way hashing function.

The user's long term key is stored in the database of KDC. When the user requests the service then KDC searches the key in Kerberos database and takes out accurate record. From the record it takes long term key.

**AS Exchange Attributes:**

The exchange consists of two messages

- KRB_AS_REQ
- KRB_AS_REP.

Figure 4.5 shows Authentication Request-Response Attributes
KRB_AS_REQ is sent from the client to Kerberos. It consists of client name realm, time parameters and authorization data. KRB_AS_REP or KRB_ERROR is sent back by the server in reply. This message consists of ticket and encrypted information.

**TGS Exchange Attributes:**

The client determines the application server’s realms and then sends the reply to TGS. Figure 4.6 shows TGS Request-Response Attributes

**AP Exchange Attributes:**
The client/server authentication (CS) exchange is used by network applications to authenticate the client to the server and vice versa. The client must have already acquired credentials for the server using the AS or TGS exchange. Figure 4.7 shows AP Request-Response Attributes.

![Diagram of AP Request-Response Attributes]

**Fig. 4.7: AP Request-Response Attributes.**

### 4.2 System Implementation components

#### 4.2.1 KDC Server - Implementation Setup

ApacheDS 2.0 Server: Using Apache Directory Studio a new server is created called as “PKLK – Apache DS 2.0.0” server.
4.2.2 Kerberos Client – Setup

Client setup – krb5.ini settings: Setting of ini file stored on client’s machine. This file has to be protected and should be stored in user’s repository folder where other users shouldn’t have access to it.