Implementation of Authentication and Transaction Security based on Kerberos

Prof R.P. Arora
Head of the Deaprtment, Computer Sc and Engg.
Dehradun Institute of Technology, Dehradun
Ms. Garima Verma
Asstt. Professor, MCA Department,
Dehradun Institute of Technology, Dehradun

***Abstract*— Kerberos is a network authentication protocol. It is designed to provide strong authentication for client/server applications by using secret-key cryptography.**

Kerberos was created by MIT as a solution to network security problems. The Kerberos protocol uses strong cryptography so that a client can prove its identity to a server (and vice versa) across an insecure network connection. After a client and server have used Kerberos to prove their identity, they can also encrypt all of their communications to assure privacy and data integrity as they go about their business.

In this paper we tried to implement authentication and transaction security in a Network using Kerberos. This project is embedded with Authentication Server application and used to derive a 64 bit key from user's password. This key is used by authentication server, to encrypt ticket granting ticket + session key. The key generated by authentication server will be used by the client at the time of transaction through the transaction server to authenticate that transaction client is valid or not.

Key Words : secret key , cryptography, authentication, ticket, session key etc.

I. INTRODUCTION

With the advent of computer the need for automated tools for protecting files and other information stored on the computer became evident [14]. This is specially the case for a shared system, such as time-sharing system, and the need is even more acute for systems that can be accessed over a public telephone network, data network, or the internet. Computer and network security is important for the following reasons [16].

• *To protect company assets:* One of the primary goals of computer and network security is the protection of company assets. By "assets," means the hardware and software that constitute the company's computers and networks. The assets are comprised of the "information" that is housed on a company's computers and networks.

• *To gain a competitive advantage:* Developing and maintaining effective security measures can provide an organization with a competitive advantage over its

competition. Network security is particularly important in the arena of Internet financial services and e-commerce.

• *To comply with regulatory requirements and fiduciary responsibilities:* Corporate officers of every company have a responsibility to ensure the safety and soundness of the organization. Part of that responsibility includes ensuring the continuing operation of the organization. Accordingly, organizations that rely on computers for their continuing operation must develop policies and procedures that address organizational security requirements.

• *To keep your job:* Finally, to secure one's position within an organization and to ensure future career prospects, it is important to put into place measures that protect organizational assets. Security should be part of every network or systems administrator's job. Failure to perform adequately can result in termination. One thing to keep in mind is that network security costs money: It costs money to hire, train, and retain personnel; to buy hardware and software to secure an organization's networks; and to pay for the increased overhead and

degraded network and system performance that result from firewalls, filters, and intrusion detection systems (IDSs). As a result, network security is not cheap.

1.1 KERBEROS

Kerberos is a network authentication protocol. It is designed to provide strong authentication for client/server applications by using secret-key cryptography [2][10]. The Internet is an insecure place. Many of the protocols used in the Internet do not provide any security. Tools to "sniff" passwords off of the network are in common use by malicious hackers. Thus, applications which send an unencrypted password over the network are extremely vulnerable. Worse yet, other client/server applications rely on the client program to be "honest" about the identity of the user who is using it. Other applications rely on the client to restrict its activities to those which it is allowed to do, with no other enforcement by the server.

Some sites attempt to use firewalls to solve their network security problems. Unfortunately, firewalls assume that "the bad guys" are on the outside, which is often a very bad assumption. Most of the really damaging incidents of computer crime are carried out by insiders. Firewalls also have a significant disadvantage in that they restrict how your users can use the Internet. (After all, firewalls are simply a less extreme example of the dictum that there is nothing more secure than a computer which is not connected to the network --- and powered off!) In many places, these restrictions are simply unrealistic and unacceptable.

Kerberos was created by MIT as a solution to these network security problems. The Kerberos protocol uses strong cryptography so that a client can prove its identity to a server (and vice versa) across an insecure network connection [13]. After a client and server have used Kerberos to prove their identity, they can also encrypt all

of their communications to assure privacy and data integrity as they go about their business.

1.1.1 Basic Concepts

The Kerberos protocol relies heavily on an authentication technique involving shared secrets [14]. The basic concept is quite simple: If a secret is known by only two people, then either person can verify the identity of the other by confirming that the other person knows the secret.

For example, let's suppose that Alice often sends messages to Bob and that Bob needs to be sure that a message from Alice really has come from Alice before he acts on its information. They decide to solve their problem by selecting a password, and they agree not to share this secret with anyone else. If Alice's messages can somehow demonstrate that the sender knows the password, Bob will know that the sender is Alice.

The only question left for Alice and Bob to resolve is *how* Alice will show that she knows the password. She could simply include it somewhere in her messages, perhaps in a signature block at the end—*Alice, Our$ecret*. This would be simple and efficient and might even work if Alice and Bob can be sure that no one else is reading their mail. Unfortunately, that is not the case. Their messages pass over a network used by people like Carol, who has a network analyzer and a hobby of scanning traffic in hope that one day she might spot a password. So it is out of the question for Alice to prove that she knows the secret simply by saying it. To keep the password secret, she must show that she knows it without revealing it.

The Kerberos protocol solves this problem with secret key cryptography. Rather than sharing a password, communication partners share a cryptographic key, and they use knowledge of this key to verify one another's identity. For the technique to work, the shared key must *symmetric*—a single key must be capable of both encryption and decryption. One party proves knowledge of the key by encrypting a piece of information, the other by decrypting it.

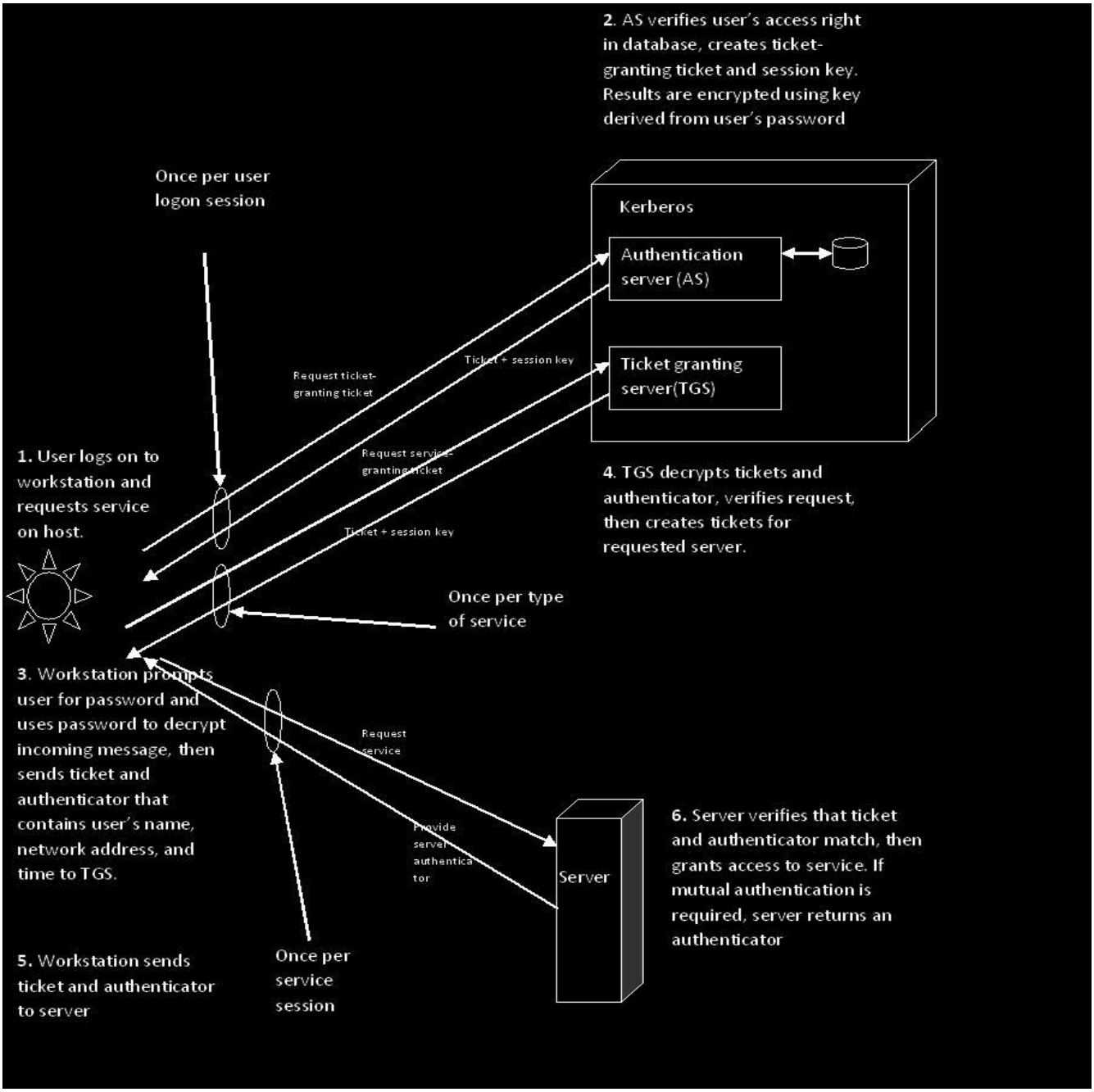


Figure 1: functional block dia. of Kerberos

2. Literature Review

K. Aruna et. al (2010), The aim of this paper is to establish a collaborative trust enhanced security model for distributed system in which a node either local or remote is trustworthy. They have also proposed a solution with trust policies as authorization semantics. Kerberos, a network authentication protocol is also used to ensure the security aspect when a client requests for certain services. In the proposed solution, they have also considered the issue of performance bottlenecks.

Steve Mallard(2010), he has defined various authentication method in order to protect the assets on your network like username and password, Biometric systems, Kerberos etc.

Dr.Mohammad N. Abdullah & May T. Abdul-Hadi (2009) they tried to establish a secured communication between the clients and mobile-bank application server in which they can use their mobile phone to securely access their bank accounts, make and receive payments, and check their balances.

Hongjun liu et. al(2008), This paper has discussed potential server bottleneck problem when the Kerberos model is applied in large-scale networks because the model uses centralized management. They have proposed an authentication model based on Kerberos,which tries to overcome the potential server bottleneck problem and can balance the load automatically

Frederick Butler , Iliano Cervesato, Aaron D. Jaggard, Andre Scedrov and Christopher Walstad (2006) Analysed Kerberos 5 protocol, and concluded that Kerberos supports the expected authentication and confidentiality properties, and that it is structurally sound; these results rely on a pair of intertwined inductions.

I. Cervesato,A. D. Jaggard,A. Scedrov,C. Walstad (2004) they presented a formalization of Kerberos 5 cross-realm authentication in MSR, a specification language based on multiset rewriting. We also adapt the Dolev-Yao intruder model to the cross-realm setting and prove an important property for a critical field in a cross-realm ticket. They also documented several failures of authentication and confidentiality in the presence of compromised intermediate realms.

2.1 Objective of the study

When we see the overall functioning of the Kerberos there are various module that need to be made for implementing Kerberos as whole for any network. For authentication of any client there is a centralized Authentication server which will generate ticket for the client using password by applying encryption technique. Simultaneously authentication server will pass a copy of ticket to the respective data server. Ticket will be unique for every data server as well as valid for only one session. Whenever client wants to perform any transaction through the server it has to send message with that ticket, the server will authenticate whether client's ticket is right or wrong, if the ticket is right it will accept the message or data sent by the client.

3. Research Methodology

To implement this project we have used Java and NetBeans 5.5 because we found Java as most suitable language to do the network programming. We have divided the whole project into three modules *client* who is a user wants to access data server, *authentication server* is the module which is used to generate ticket and return back to the client who accesses data server so that data server can easily check whether the client who is coming with ticket is correct or not, and *data server* is the site where data is stored and can be utilized by the clients. We have used the concept of socket programming to implement client, authentication server and data server.

For the generation of ticket from the authentication server we have used Data Encryption Standard (DES) in authentication server which will use 64 bit plain text and 56 bit key.

4. Implementation

The whole project is divided into three modules- Client site, authentication server and data server.

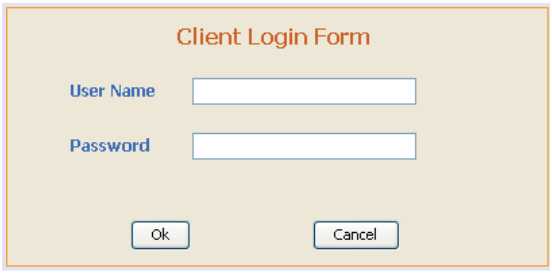
4.1 Client Module

Client is any user who can apply to any data server for service. The obvious security risk is that of impersonation. An opponent client can pretend to be another client and obtain unauthorized privileges on the data server sites. To counter this threat, data servers must be able to confirm the identities of clients who request service. We followed following steps:

- Client will logon to its own terminal by using user name and password. These user and passwords and predefined and assigned to every client on the network. Every client have unique user name with two passwords, one password is used to logon to the client terminal and another is called as transaction password which he will submit to the authentication server.

Code Section

```
Class.forName("com.mysql.jdbc.Driver").newInstance()
;

Connection con =
DriverManager.getConnection("jdbc:mysql://localhost/t
est?"+"user=root&password=garima");


Statement stmt=con.createStatement();
ResultSet rs=stmt.executeQuery("select * from user
where userid='"+usertxt.getText()+"'");

String  u,p;


rs.next();


u=rs.getString(1);

p=rs.getString(2);


if(p.compareTo(passtx.getText())!=0)

{


JOptionPane.showMessageDialog(this,"Password
Wrong");

}
```



- After the successful login client will submit his details with transaction password to the authentication server. Details include – username, transaction password and name of the data server.

- Again entered transaction password will be checked into the client database then finally sent to the authentication server.

Code Section

```
Class.forName("com.mysql.jdbc.Driver").newInstan
ce();
```

```
Connection con =
DriverManager.getConnection("jdbc:mysql://localh
ost/test?"+ "user=root&password=garima");

Statement stmt=con.createStatement();

ResultSet rs=stmt.executeQuery("select * from
user where userid='"+txtuser.getText()+"'");

String  u,p;

 if(rs==null)

{
JOptionPane.showMessageDialog(this,"User
Name is Wrong");

}

rs.next();

//writing data to authentication
server u=rs.getString(1);

 p=rs.getString(3);

 if(p.compareTo(txtpas.getText())!=0)

{
JOptionPane.showMessageDialog(this,"Password
Wrong");

}

else

{

        try

       {

            Socket clientSocket = new
Socket("192.168.10.133", 6789);
PrintStream ps=new
PrintStream(clientSocket.getOutputStream());

DataInputStream dis=new
DataInputStream(clientSocket.getInputStream());

                ps.println(u);

                ps.println(p);

                ps.println(serverip.getText());

                s=dis.readLine().toString();


JOptionPane.showMessageDialog(this,s);
//  clientSocket.close();

        }catch(Exception e)

            {
        JOptionPane.showMessageDialog(this,e.t
        oString()+" :he");

            }

  }

    }catch(Exception e)

            {


JOptionPane.showMessageDialog(this,"User
Name is Wrong");

            }

  }
```

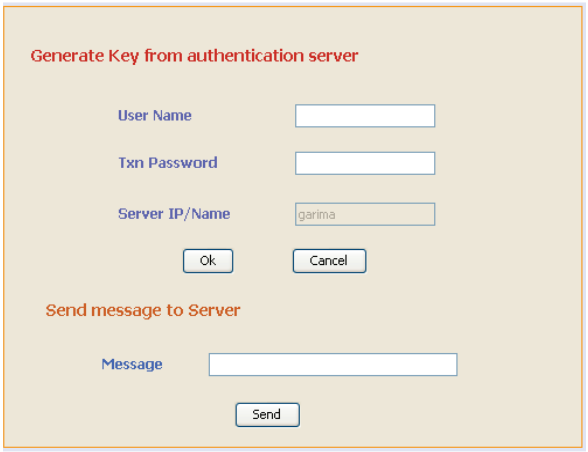
- After receiving ticket from the authentication server the client will send message + ticket to the data server.

Code section

```
try

{

 clientSocket2 = new Socket("192.168.10.216",
7211);

 JOptionPane.showMessageDialog(this,"sending");

PrintStream pserver=new
PrintStream(clientSocket2.getOutputStream());

DataInputStream diserver=new
DataInputStream(clientSocket2.getInputStream());
```

// sending message to data server

```
pserver.println(servermsg.getText());

pserver.println(s);


JOptionPane.showMessageDialog(this,diserver.readLine());

clientSocket2.close();


}catch(Exception e){}
```

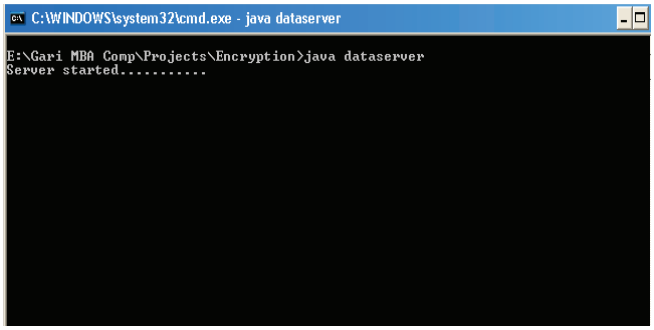


4.2 Authentication Server Module

Authentication server is central authorities that knows the passwords of all clients and store these in a centralized database. In addition, the AS shares a unique secret key with each server [14]. These keys have been distributed physically or in some other secure manner. For example – the user logs on to a workstation and requests access to server V: the client module C in the user's workstation requests the user's password and then sends a message to AS that includes the user's ID, server's ID and user's password. The AS checks its database to see if the user has supplied the proper password for this user ID and whether this user is permitted access to server V. If both tests are passed, the AS accepts the user as authentic and creates a Ticket. This ticket is then sent back to C. For the encryption we have used DES algorithm.

The Following steps are included in this module:

- After Start of authentication server as well as data server



After starting the authentication server it can accept any request coming to its port address from client side.

Code Section

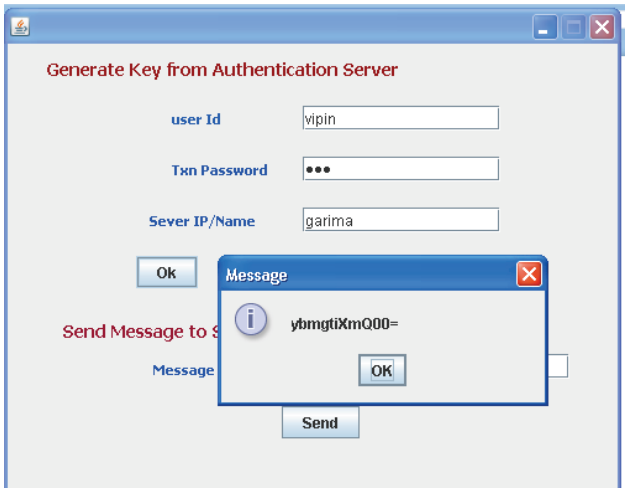
```
welcomeSocket = new ServerSocket(6789);

connectionSocket = welcomeSocket.accept();
```

- Generation of ticket using DES algorithm and store a copy of ticket in its own database as well as send a copy on server site.

Code Section

```
key =
KeyGenerator.getInstance("DES").generateKey();
```

```
System.out.println("Client :"+ connectionSocket);

PrintStream pserver=new
PrintStream(cs.getOutputStream());

PrintStream ps=new
PrintStream(connectionSocket.getOutputStream());

DataInputStream dis=new
DataInputStream(connectionSocket.getInputStream());

clientname=dis.readLine();

clientSentence=dis.readLine();

servername=dis.readLine();

DesEncrypter ds=new
DesEncrypter(key,clientSentence);

String enc= ds.encrypt(clientSentence);

 pstm.setString(1,clientname);

 pstm.setString(2,enc);

 pstm.setString(3,servername);

pstm.executeUpdate();


pstm1.setString(1,clientname);

pstm1.setString(2,enc);

pstm1.executeUpdate();

ps.println(enc);
```



4.3 Data Server Module

Now after receiving ticket client can now apply to Server for service. Client send a message to server containing its ID and ticket. Server decrypts the ticket and match it with ticket stored in the database. If these two match, the server considers the user authenticated and requested service.

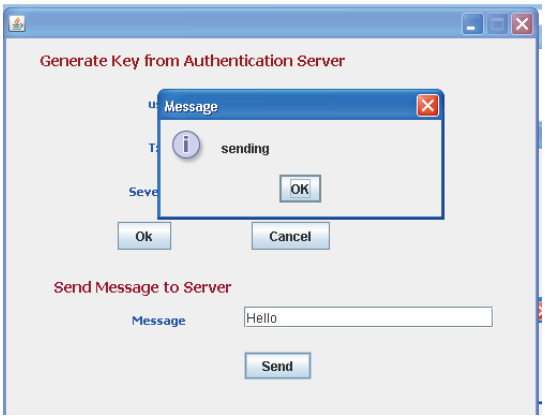
Following steps we have included in this module-

- Client will send a message with ticket to the data server after receiving ticket from authentication server.

Code Section

```
clientSocket2 = new Socket("192.168.10.216",
7211);

JOptionPane.showMessageDialog(this,"sending");

PrintStream pserver=new
PrintStream(clientSocket2.getOutputStream());

DataInputStream diserver=new
DataInputStream(clientSocket2.getInputStream());

pserver.println(servermsg.getText());

pserver.println(s);
```

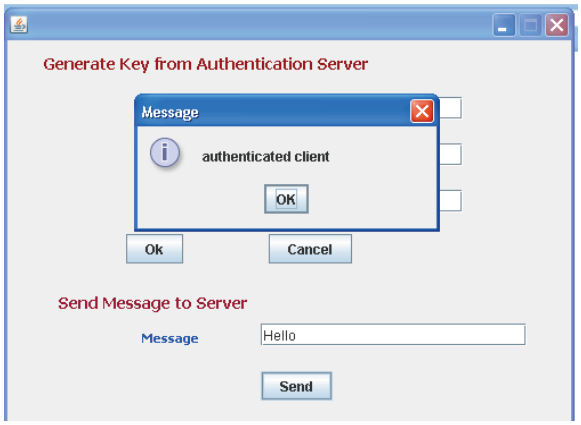
```
JOptionPane.showMessageDialog(this,diserver.readLi
ne());

        clientSocket2.close();
```



- Now the data server will verify the ticket, after verification the data server will send a message to client whether he is authentic or not.

Code Section

```
msg=dclient.readLine();

k1=dclient.readLine();

ResultSet rs=stmt.executeQuery("select * from dser
where keyid='"+k1+"'");

if(rs!=null)

ps.println("authenticated client");

else

            ps.println(“Not authorized”);

System.out.println(msg);

new job1(consoc);

ps.close();
```



5. Conclusion

Authentication is critical for the security of computer systems. Without knowledge of the identity of a principal requesting an operation, it is difficult to decide whether the operation should be allowed. Traditional authentication methods are not suitable for use in computer networks where attackers monitor network traffic to intercept passwords. The use of strong authentication methods that do not disclose passwords is imperative. The Kerberos authentication system is well suited for authentication of users in such environments.

If we talk about the unprotected environment, any client can apply to any server for service. This has a security risk of impersonation. An opponent can pretend to be another client and obtain unauthorized privileges on server machines. In the above scheme the transaction will be highly secured in the sense that Authentication server creates a ticket which is further encrypted using the secret key shared by the server and authentication Server. This ticket then sends back to client. Because the ticket is encrypted, it cannot be altered by client or by an opponent.

References

[1] K. Aruna et. al (2010), "A new collaborative trust enhanced security model for distributed systems". International Journal of Computer Application, No-26
[2] Steve Mallard(2010), "Methods of authentication", Bright Hub
[3] Hongjun liu et. al(2008), "A distributed expansible authentication model based on Kerberos" Journal of Network and Computer Application, Vol.31, Issue 4
[4] Dr.Mohammad N. Abdullah & May T. Abdul-Hadi, "A Secure Mobile Banking Using Kerberos Protocol", Engg & Technology Journal, Vol 27, No 6, 2009.
[5] "How Kerberos Authentication Works", Network on line magazine, Jan 2008
[6] "How Kerberos Authentication Works",Learn Networking on line magazine, Jan'2008
[7] Frederick Butler, Iliano Cervesato, Aaron D. Jaggard, Andre Scedrov and Christopher Walstad "Formal Analysis of Kerberos 5", Sep 2006
[8] Rong Chen, Yadong Gui and Ji Gao, "Modification on Kerberos Authentication Protocol in Grid Computing Environment", vol 3032, 2004.
[9] I. Cervesato,A. D. Jaggard,A. Scedrov,C. Walstad, "Specifying Kerberos 5 cross-realm authentication",vol 3032, 2004.
[10] "Security of Network Identity: Kerberos or PKI", System News (2002), Vol.56, Issue-II
[11] Ian Downnard, "Public-key cryptography extensions into Kerberos". IEEE Potentials 2002.
[12] B. Clifford Neuman and Theodore Ts'o, Kerberos: An Authentication Service for Computer Networks, IEEE Communications 32 (1994), no. 9, 33--38.
[13] MIT Kerberos Website, " http://web.mit.edu/kerberos/www".
[14] William Stallings, "Cryptography and Network Security", Third Edition.
[15] Ravi Ganesan, "Yaksha' : Augmenting Kerberos with Public Key cryptography".
[16] John E. Canavan, "Fundamentals of Network Security".
[17] Chris Brenton with Cameron hunt , " ACTIVE DEFENCE A comprehensive guide to network security"