CHAPTER 3

ENHANCED KERBEROS SECURITY:
An application of the proposed system
3.1 Introduction

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

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 systems crackers. 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 [62]. 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 our 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 is an authentication protocol, and at the same time a Key Distribution Center (KDC), that has become very popular [63]. Several systems including Windows 2000 use kerberos. Kerberos is named after the three-headed dog in Greek mythology that guards the gates of Hades. Originally designed at MIT, it has gone through several versions. It was developed as a part of Project Athena at MIT to provide a solution to network security problems [64]. Consider a distributed environment having many users on different workstations and services, available on servers distributed across the network. An unauthorized user may be able to gain access to services and data that he or she is not authorized to access. Instead of building elaborate authentication protocols at each server, kerberos provides a centralized authentication server, whose function is to authenticate users to servers and servers to users.
Kerberos 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 their communications to assure privacy and data integrity, as they go about their business.

**Kerberos involves three servers** in addition to the client workstation as shown in figure 3.1: an **authentication server (AS)**, a **ticket-granting server (TGS)**, and a real (data) **server (DS)** that provides services to others [65]. In our example, Bob is the real server, Alice is the user/client server and Eve is the intruder.

**Authentication server (AS):** It verifies the users during the login process. It stores a secret password for every user. The AS is the KDC in the kerberos protocol. Each user registers with the AS and is granted a user identity and a password. The AS has a database with these identities and the corresponding passwords. The AS verifies the user, issues a session key to be used between Alice and the TGS, and sends a ticket for the TGS.

**Ticket granting server (TGS):** It issues 'proof of identity tickets'. These tickets are used to tell the other servers that the bearer of the TGS ticket is actually the person who he or she claims to be. The TGS issues a ticket for the real server. It also provides the session key \(k_{AB}\) between Alice and Bob. Kerberos has separated the user verification from ticket issuing. In this way, although Alice verifies her ID just once with AS, she can contact TGS multiple times to obtain tickets for different real servers.

**The Server:** This is the server that does the work the clients want to be performed. The real server (Bob) provides services for the user (Alice). Kerberos is designed for a client-server program such as FTP, in which a user uses the client process to access the server process.
3.2 Operation in kerberos systems

A client process (Alice) can receive a service from a process running on the real server (Bob) in six steps [66].

Step 1

Alice sends her request to AS in plaintext, using her registered identity.

Step 2

The AS sends a message encrypted with Alice’s symmetric key $K_A$. The message contains two items: a session key $K_S$ that is used by Alice to contact TGS and a ticket for TGS that is encrypted with the TGS symmetric key $K_{TG}$. Alice does not know $K_A$, but when the message arrives, she types her password. The password and the appropriate algorithm together create $K_A$ if the password is correct. The password is then immediately destroyed; it is not sent to the network, and it does not stay in the terminal. It is only used for a moment to create $K_A$. The process now uses $K_A$ to decrypt the message sent; $K_S$ and the ticket are extracted.

Step 3

Alice now sends three items to the TGS. The first is the ticket received from AS. The second is the name of the real server (Bob), and the third is a timestamp which is encrypted by $K_S$. The timestamp prevents a replay by Eve.

Step 4

Now, TGS sends two tickets, each containing the session key between Alice and Bob $K_{AB}$. The ticket for Alice is encrypted with $K_S$; the ticket for Bob
is encrypted with Bob's key $K_B$. She cannot replay step 3 because she cannot replace the time-stamp with a new one (she does not know $K_S$). Even if she is very quick and sends the step 3 messages before the time-stamp has expired, she still receives the same two tickets that she cannot decipher.

**Step 5**

Alice sends Bob's ticket with the time-stamp encrypted with $K_{AB}$.

**Step 6**

Bob confirms the receipt by adding number one to the time-stamp. The message is encrypted with $K_{AB}$ and sent to Alice.

**Requesting and receiving services**

After step 6, Alice can request and receive services from Bob using $K_{AB}$ as the symmetric shared key.

**Using different servers**

If Alice needs to receive services from different servers, she needs to repeat only the last four steps. The first two steps have verified Alice's identity and need not be repeated. Alice can ask the TGS to issue tickets for multiple servers by repeating step 3 to 6.

### 3.3. Kerberos version 5

The minor difference between version 4 and version 5 are briefly listed below [67].

1. Version 5 has a longer ticket lifetime.
2. It allows tickets to be renewed.
3. It can accept any symmetric-key algorithm.
4. It uses a different protocol for describing data types.
5. It has more overhead than version 4.

**Realms**

Kerberos allows the global distribution of ASs and TGSs, with each system called a realm. A user may get a ticket for a local server or a distant server. In the second case, for example, Alice may ask her local TGS to issue a ticket that is accepted by a distant TGS. The local TGS can issue this ticket if the distant TGS is registered with the local one. Then Alice can use the distant TGS to access the distant real server.
3.4 Kerberos Vs. Secure Sockets Layer (SSL)

SSL has two major advantages over kerberos: (1) It doesn't require an accessible trusted third party; (2) it can be used to establish a secure connection even when one end of the connection doesn't have a "secret key" or "password". These two advantages make it ideal for secured web communication and for similar applications where there is a large user base, which is not known in advance.

Here are some disadvantages of SSL:

1) **Key revocation.** If a Verisign certificate issued to a user is compromised and must be revoked, how will all the servers with whom that user interacts know that the certificate is no longer valid? Either revocation certificates have to be circulated to all relevant servers and cached for a long time, or servers have to verify incoming user certificates against a "revocation server." In that case, the revocation server must be a highly available third party, which means we have eliminated one of the two major advantages of SSL over kerberos. Kerberos principals can be disabled at will on the KDC and will then become unusable as soon as any cached tickets expire, on the order of hours, without any action by servers.

2) **Key security.** If we issue a Verisign certificate, it has to live on the hard disk. Yes, it may be encrypted there such that we have to unlock it with a password before we can use it, but it's still on the hard disk and therefore vulnerable to cracking attacks. On the other hand, we don't need any sort of certificate to authenticate to kerberos -- all we need is the password, which is in the brain, not on a hard disk.

3) **Cost of use.** Kerberos doesn't infringe on any patents. Which means that it can be used for free, while SSL users may have to pay.

4) **Open standards.** Kerberos has been free from the beginning. The standards documenting it are open and have been developed openly from the start. On the
other hand, SSL was developed by a company with a commercial interest in ensuring that its standards become THE standard.

5) **Flexibility.** Kerberos is more flexible than SSL. For example, if we want to add a new authentication technology to kerberos, all we have to do is modify KDC and the ticket-acquiring client to know how to do the new authentication. Then, it can be used to get kerberos tickets, which will look the same as any other kerberos tickets and will be usable with any kerberos-capable application. On the other hand, if we want to implement a new authentication technology for SSL, we would have to get new versions of all the SSL-capable applications.

### 3.5. Shortcomings of kerberos

Kerberos makes no provisions for host security; it assumes that it is running on trusted hosts with an untrusted network. If the host security is compromised, then kerberos is compromised as well [68]. However, the degree to which kerberos is compromised depends on the host that is compromised. If an attacker breaks into a multi-user machine and steals all of the tickets stored on that machine, he can impersonate the users who have tickets stored on that machine but only until those tickets expire.

Kerberos uses a principal's password (encryption key) as the fundamental proof of identity. If a user's kerberos password is stolen by an attacker, then the attacker can impersonate that user with impunity.

Since the KDC holds all of the passwords for all of the principals in a realm, if host security on the KDC is compromised, then the entire realm is compromised.

In kerberos, authenticators are valid for 5 minutes. If an attacker sniffs the network for authenticators, they have a 5 minutes window in which they can re-use it and gain access to the same service we used [68].

Since anybody can request a TGT for any user, and that ticket is encrypted with the user's secret key (password), it is simple to perform an offline attack on this ticket by trying to decrypt it with different passwords.
3.5.1 Double encryption

The tickets provided to the clients are encrypted twice, once with the secret key of the target server and again with the secret key known only to the client [68]. The second encryption is unnecessary and only puts more loads on the computational resources.

3.5.2 Session keys

Each ticket includes a session key, used by the client to encrypt the authenticator sent to the service, associated with that ticket. The same ticket is used repeatedly by the client to gain service from a particular server. This increases the risk of messages, from an old session to the client or the server, being replayed by an intruder.

3.5.3 Password attacks

The message from AS to the client includes data encrypted with a key, based on the client password. This can be captured by an opponent snooping on the network who may attempt to decrypt it by using various passwords. Kerberos is vulnerable to password guessing attacks, since it assumes users will not select poor passwords, such as words that can be found in a dictionary, common names, user's name or initials [68]. If an attacker successfully guesses the user's password, he will be able to impersonate him to any verifier. If the decryption is successful the opponent discovers the client's password and can use it to gain authentication credentials from kerberos.

3.5.4 Encryption algorithm in kerberos

In kerberos we prove our identity by being able to decrypt or encrypt data using an encryption key that we share with the KDC. However, a 56-bit DES key is hard for humans to remember. So, whenever a person enters in their "kerberos password", it is really converted to an encryption key by a function called string2key(). This function converts the plaintext password via a one-way hash algorithm to an encryption key. In kerberos 4 this is always a DES key. In
kerberos 5 it could be a key for algorithms other than DES (but currently DES is still the most widely used algorithm in kerberos 5).

The string2key() function takes an optional argument called the key salt. This is an additional input to the one-way hash algorithm. If a salt is supplied, it is concatenated to the plaintext password and the resulting string is converted using the one-way hash algorithm.

In kerberos 4, a salt was never used. The password was the only input to the one-way hash function. This has a serious disadvantage; if a user happens to use the same password in two kerberos realms, a key compromise in one realm would result in a key compromise in the other realm.

In kerberos 5 the complete principal name (including the realm) is used as the salt. This means that the same password will not result in the same encryption key in different realms or with two different principals in the same realm.

The MIT kerberos 5 KDC stores the key salt algorithm along with the principal name, and that is passed back to the client as part of the authentication exchange. This means that if we convert our kerberos database from kerberos 4 to kerberos 5, kerberos 5 clients can use the correct string2key algorithm to convert our password to the matching encryption key.

It's worth pointing out that this is only an issue for the cases when we need to convert a plaintext password to an encryption key. Programs that deal directly with encryption keys never deal with plaintext passwords, and as a result this is not an issue with them.

3.6. Proposed system for kerberos

Having kerberos protocol, the server can be assured offering services to the correct client who is entitled to have access. This is because kerberos assumed that only the correct user could use the credential, as others do not have the password to decrypt it. And also because of this, a user can actually request the credential of others. That is, the user is not authenticated at the beginning stage.

In this way, an attacker can obtain the credential of another user, and perform off-line attack by using a password guessing approach as the ticket is sealed only by the password. The kerberos authentication system has such a security weakness.
This problem can be solved by the proposed system in the following ways.

1. All the encryptions could be performed using the proposed multilevel cryptographic algorithm. Since the current kerberos system uses a standard symmetric key encryption algorithm DES, it is easy for an intruder to find out the key and decrypt. But when the proposed multilevel system is used, only the authorized persons, who have the decryption algorithm, could decrypt the encrypted text. Any other intruder, who wants to perform off-line attack, will not be able to do so because this algorithm protects the message in a much stronger way using password controlled variable block cipher with cipher block chaining mode. It is very difficult to decrypt the message even with the algorithm available. Because this algorithm gives an extra layer of protection with a password. The chances of password guessing approach for any intruder are nullified because the proposed system does not store the password anywhere in the system. Hence no attempt can be made to find it out.

2. By integrating the proposed system with the smart card technology, some of the kerberos system problems can be overcome. The whole idea is to enhance the security of kerberos authentication by authenticating the user directly at the beginning and before the granting of the initial ticket, so that one user cannot have the ticket of another. And, the use of smart card requires user logging into the system not only by recalling a password, but also to be in possession of a token.

3. Another way to enhance security is to use biometric technology with the proposed system in the smart card. Biometrics information of the cardholder can be placed on the card, so that the smart card can corporate with biometrics scanner to authenticate the user directly at the first stage of processing. Before granting the initial ticket, this authentication could take place, to avoid any intruder to pretend as the cardholder. The proposed multilevel approach, which combines the techniques of cryptography and steganography, could be applied to embed the biometrics information of the cardholder into his photograph in the smart card. Since this algorithm provides a robust protection to the information against attacks, the biometrics details could not be easily tapped by any fraudulent.
3.7. Results and discussion
3.7.1 Cryptanalysis of the proposed system

We tried DES encryption algorithm, which is used in kerberos and the proposed multilevel algorithm on text files with redundant data of various sizes. We then applied cryptanalysis on the cipher text files created by these two methods. We estimated the time taken for cryptanalysis.

Since the existing system employs simple DES, the time taken for cryptanalysis is very low [69,70]. But the proposed multilevel system applies the cipher block-chaining mode along with variable block cipher and hence it nullifies the redundancy in the resultant cipher text. This makes the cryptanalysis process difficult. The results are shown in figure 3.2.

![Analysis of time for Cryptanalysis of redundant data](image)

**Figure 3.2 Comparison of time for cryptanalysis of existing kerberos and the proposed systems for redundant data**

Next, cryptanalysis was performed on the cipher text files created by the existing DES and the proposed multilevel system for non-redundant text data. The existing system takes more time for cryptanalysis for non-redundant data than for redundant data [71,72]. But the proposed system takes the same time for cryptanalysis for both redundant and non-redundant data. But in both the cases, the proposed multilevel system takes more time for cryptanalysis than the existing system. The following figure 3.3 shows the cryptanalysis for non-redundant data.
3.7.2 Steganalysis of the proposed system

The steganography algorithm proposed in the system inserts information and manipulate the images in ways as to remain invisible to the human eye. However, any manipulation to the image introduces some amount of distortion and degradation of some aspect in the "original" image's properties. Without knowing which algorithm is used and how the insertion was done, if any, stegokey is used, detecting the hidden information may become quite complex.

To begin evaluating images for additional, hidden information, the concept of defining a "normal" or average image was deemed desirable. Defining a normal image is somewhat difficult when considering the possibilities of digital photographs, paintings, drawings, and graphics. Only after evaluating many original images and stego-images as to color composition, luminance, and pixel relationship do anomalies point to characteristics that are not "normal" in other images. Several patterns became visible when evaluating many images used for applying steganography. The chosen message and known cover attacks were quite useful in detecting these patterns. In images that have color palettes or indexes, colors are typically ordered from the most used colors to the least used colors to reduce table lookup time. The changes between color values may change gradually but rarely, if ever, in one bit shifts. Gray-scale image color indexes do shift in 1-bit increments, but all the RGB values are the same. Applying a similar
approach to monochromatic images other than gray-scale, normally two of the
RGB values are the same with the third generally being a much stronger saturation
of color. Some images such as hand drawings, fractals and clip art may shift
greatly in the color values of adjacent pixels. However, having occurrences of
single pixels outstanding may point to the existence of hidden information.

Added content to some images may be recognizable as exaggerated noise.
This is a common characteristic for many bit-wise tools as applied to 8-bit images.
Using 8-bit images without manipulating the palette will, in many cases, cause
color shifts as the raster pointers are changed from one palette entry to another. If
the adjacent palette colors are very similar, there may be little or no noticeable
change. However, if adjacent palette entries are dissimilar, then the noise due to
the manipulation of the LSBs is obvious. For this reason the proposed system
stresses the use of gray-scale images (those with 256 shades of gray). Gray-scale
images are special occurrences of 8-bit images and are very good covers because
the shades gradually change from color entry to color entry in the palette.

Using images with vastly contrasting adjacent palette entries will foil the
proposed steganography software because small shifts to the LSBs of the raster
data will cause radical color changes in the image that advertise the existence of a
hidden message. Without altering the 8-bit palette, changes to the LSBs in the
raster data may show dramatic changes in the stego-image.

3.8. Conclusion

Kerberos proves to be the best authentication system for networks in a
distributed environment. But in order to gain access to the real server, the client
has to be authorized by two more servers. This imposes overhead to the entire
network. Kerberos is vulnerable to password guessing attacks, since it assumes
users will not select poor passwords, such as words that can be found in a
dictionary, common names, user's name or initials. If an attacker successfully
guesses the user's password, he will be able to impersonate him to any verifier.

Also, kerberos has no control over the workstations or machines where the
user is entering his password. It assumes that an attacker has no opportunity to
position himself between the user and the client to obtain the password, and
cannot monitor the path between the user and the initial authentication program.
Kerberos will only protect messages from software that has been written or modified to use it; it will not protect all messages between two computers. For instance, if a person authenticates himself to kerberos and then uses a telnet program without encryption, his telnet session will not be protected. It is necessary to use a kerberos enabled telnet program with the encryption option on in order to have a fully protected session.

With the recent developments in the field of data security, these additional servers could be easily replaced by any of the alternates proposed here. By applying the proposed multilevel system to kerberos as suggested above, its security could be enhanced.