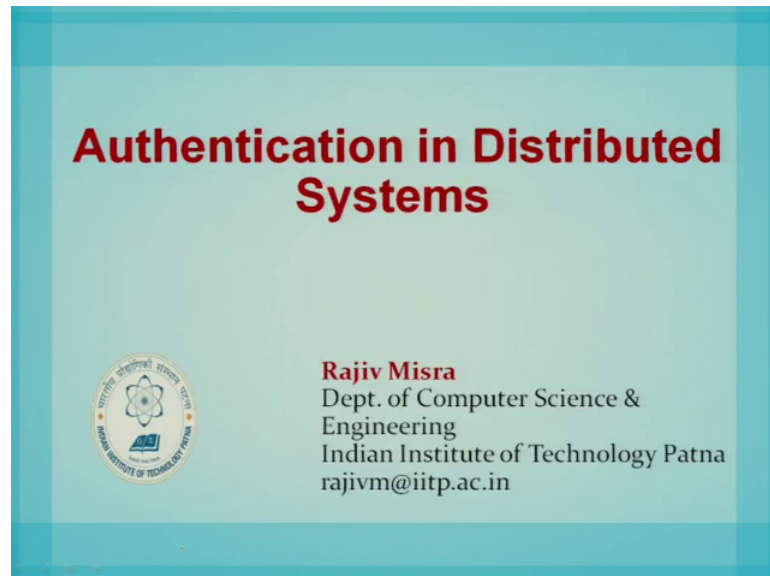


Distributed Systems
Dr. Rajiv Misra
Department of Computer Science and Engineering
Indian Institute of Technology, Patna

Lecture - 25
Authentication In Distributed Systems

(Refer Slide Time: 00:13)



Authentication in distributed systems.

(Refer Slide Time: 00:16)

The slide has a light green background with a dark green border. The title "Introduction" is in a bold red font. Below it, there are five bullet points. The first bullet point is "A **distributed system** is susceptible to a variety of **security threats**." with handwritten notes "entities - client-server, host, AWS, etc" written in red above it. The second bullet point is "A principal can impersonate other principal and authentication becomes an important requirement." The third bullet point is "Authentication is a process by which one principal verifies the identity of other principal." with handwritten notes "client to server" and "mutual authentication" written in red above it. The fourth bullet point is "In **one-way authentication**, only one principal verifies the identity of the other principal." The fifth bullet point is "In **mutual authentication**, both communicating principals verify each other's identity."

Introduction distributed system is susceptible to a variety of security threats. A principal can impersonate other principal and authentication becomes an important requirement. So, principal means the entities which are participating or communicating with each other for example, client server both can be the entities or the host, machines and the users etcetera they are called principals. So, principals can impersonate other principals and the authentic authentication becomes an important requirement in this particular susceptible scenario in a distributed system.

So, authentication is a process by which one principal verifies the identity of other principal. For example, the client, the client can verify about the identity of the server whether it is that particular server with whom it is intended to it is communicating. Similarly, server can also want to verify the identity whether it is the client with whom it is going to communicate. So, in this particular scenario it is called mutual authentication.

So, authentication is a process by which one principal verifies the identity of the other principal, both the principals they want to verify their identities then it is called a mutual authentication otherwise it is one way authentication. So, one way authentication only one principal verifies the identity of the other principal for example, if the client want to verify whether it is that particular specific server or identity of that server then it is one way authentication or server want to identify the identity of the client mutual authentication both the communicating principles verify each other that I explained.

So, the background and definitions authentication is a process of verifying that principles identity is claimed so there are two things here one is the verification the other is the identity.

(Refer Slide Time: 03:04)

Background and definitions

- **Authentication** is a process of verify that the principal's identity is as claimed. (Identification, verified) → authentication
- Authentication is based on the possession of some **secret information, like password**, known only to the entities participating in the authentication.
- When an entity wants to authenticate another entity, the former will verify if the latter possesses the knowledge of the secret.

So, identification of a principal and then as claimed the identity it has to be verified, when both these things are satisfied then it is called authentication.

So, again I am reading it because both the terms are there in this particular definition of authentication. Authentication is a process of verifying that the principals identity is as claimed. So, authentication is based on position of some secret information in the distributed system like password known only to the entities participating in the authentication, when an entity want to authenticate another entity the farmer will verify if the letter possesses the knowledge of that secret information like password etcetera.

A simple classification of authentication protocol so classified based on the cryptographic technique which are used the authentication protocols are listed as follows.

(Refer Slide Time: 04:16)

A simple classification of authentication protocols

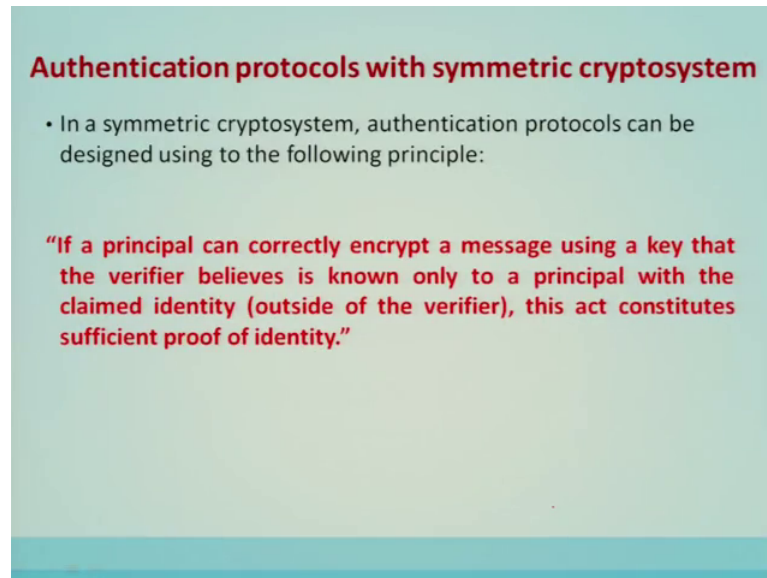
- Classified based on **the cryptographic technique** used.
- There are two basic types of cryptographic techniques: **symmetric ("private key") and asymmetric ("public key")**.
- **Symmetric cryptography** uses a single private key to both encrypt and decrypt data. (Let $\{ X \}_k$ denote the encryption of X using a symmetric key k and $\{ Y \}_{k^{-1}}$ denote the decryption of Y using a symmetric key k .)
- **Asymmetric cryptography**, also called Public-key cryptography, uses a secret key (private key) that must be kept from unauthorized users and a public key that is made public. (pair of keys - Secret key, Public key)
- Data encrypted with the public key can be decrypted only by the corresponding private key, and data signed with the private key can only be verified with the corresponding public key.

There are 2 kinds of cryptographic techniques symmetric key cryptography that is called also properly known as a private key, the other cryptography technique is called asymmetric key cryptography and that is also called a public key. So, based on two type of cryptography techniques which is used, we will see the classification of authentication protocol later on.

So, first we will classify the cryptographic techniques based on these two techniques. So, the first type of cryptography is called symmetric cryptography, it uses a single private key to both encrypt and decrypt the data and asymmetric cryptography is called public key cryptography, that uses the secret key that must be kept secret from unauthorized users and another key that is also used called public which is made public for the encryption.

So, data encrypted with the public key can be decrypted only by that corresponding private key. So, it is a pair of keys involved in a symmetric cryptography that is a pair of keys that is the secret key and a public key, together is basically the keys which are used in asymmetric cryptography. So, data is encrypted with the public key can be decrypted only by the corresponding private key and the data signed with the private key can only be verified with the corresponding public key.

(Refer Slide Time: 06:28)



Authentication protocols with symmetric cryptosystem

- In a symmetric cryptosystem, authentication protocols can be designed using to the following principle:

“If a principal can correctly encrypt a message using a key that the verifier believes is known only to a principal with the claimed identity (outside of the verifier), this act constitutes sufficient proof of identity.”

So, in symmetric crypto graph crypto system authentication protocol can be designed using the following principle, if a principal can correctly encrypt a message using the key that the verifier believes is known only to the principal with the claimed identity this act constitutes sufficient proof of identity.

So, let us go ahead for the case studies.

(Refer Slide Time: 06:48)

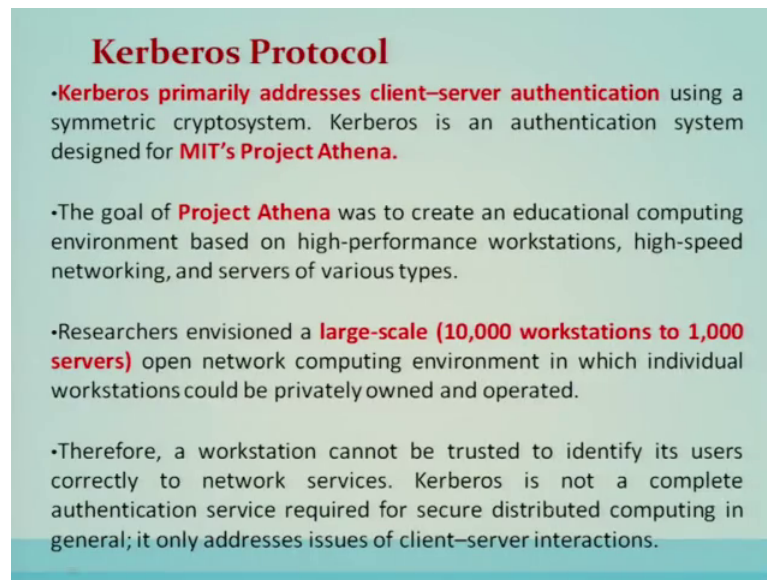


Case Studies:

1. **KERBEROS PROTOCOL**
2. **SECURE SOCKET LAYER (SSL) PROTOCOL**

So, in this lecture we are going to see two different case studies one is called Kerberos protocol the other is called secure socket layer protocol, Kerberos protocol.

(Refer Slide Time: 07:02)



Kerberos Protocol

- Kerberos primarily addresses client–server authentication using a symmetric cryptosystem. Kerberos is an authentication system designed for MIT's Project Athena.
- The goal of Project Athena was to create an educational computing environment based on high-performance workstations, high-speed networking, and servers of various types.
- Researchers envisioned a large-scale (10,000 workstations to 1,000 servers) open network computing environment in which individual workstations could be privately owned and operated.
- Therefore, a workstation cannot be trusted to identify its users correctly to network services. Kerberos is not a complete authentication service required for secure distributed computing in general; it only addresses issues of client–server interactions.

So, Kerberos primarily addresses client server authentication using symmetric cryptography, Kerberos is an authentication system designed for MIT project Athena. The goal of the project athena was to create an educational computing environment based on high performance workstations, high speed networks and the servers of various types. Researchers envisioned a large scale 10000 workstation to 1000 servers, open network computing environment, in which individual workstation could be privately owned and operated therefore, the workstation cannot be trusted to identify its user correctly to the network services. Kerberos is not a complete authentication required for secure distributed computing in general it only addresses the issues of client server interactions.

Here we will discuss Kerberos authentication protocol.

(Refer Slide Time: 08:06)

Contd...

- Here, we will describe the **Kerberos authentication protocol**.
- Kerberos' design is based on the use of a **symmetric cryptosystem together with trusted third-party authentication servers**.
- The basic components include:
 - (i) Authentication Servers (Kerberos servers) and
 - (ii) Ticket-Granting Servers (TGSs).

The diagram shows a client-server relationship where the password is exposed on the network. The client sends a password to the servers, which are connected to a network. The servers provide services like payment and file services. A replay attack is shown where the password is intercepted and used again.

Kerberos design is based on the use of symmetric cryptosystem together with the trusted third party authentication server, before we go ahead let us see where this particular Kerberos system is useful in a distributed system. So, the Kerberos is primarily used in the distributed system, in the distributed systems you know that there are the client the servers. So, if a client want to take the services which are offered by the servers maybe there are different type of services which are available in this mode or a distributed system for example, the payment system and the services like file services and so on.

So, if a client want to access to these services which are provided through different servers. So, the most important part is that the client has to produce its credentials in the form of a password. So, the password has to flow through the network and you know that this network is unsafe as far as if the password is flowing openly on the network; even if it is encrypted the each dropper can tap this particular password and can replay it again. So, there are various issues involved that how can these services can be offered without these particular passwords to be flown on the network for this a Kerberos authentication protocol is there which will provide the authentication between client and server, without exposing the passwords on the network.

So, you will see in this particular protocol in the discussion that the client and server they are authenticated with the help of passwords, but the passwords will not flow on the network and yet the authentication is completed. So, that particular protocol is designed

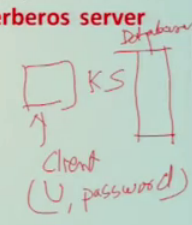
with that particular specific use, it is designed it is nowadays has become a standard and this will be used everywhere in most of the systems which are operating in the network that is in the form of distributed systems. So, this Kerberos authentication protocol has 2 components. So, the first component is called the authentication server or it is also called Kerberos servers, the other component here which will provide the authentication is called ticket granting server or TGS.

So, we will see that how using these Kerberos authentication server and using ticket granting server it can authenticate between client and server without exposing the password to flow on the, on the network. So, there is a process which is called initial registration where the each client registers with the with the Kerberos server.

(Refer Slide Time: 11:56)

Initial registration

- Every client/user registers with the Kerberos server by providing its **user i.d., U**, and a **password, password**.
- The Kerberos server computes a **key $ku = f(\text{password})$** using a **one-way function f** and stores this key in a database.
- Note that **ku is a secret key** that depends on the password of the user and is shared by **client U** and the **Kerberos server** only.

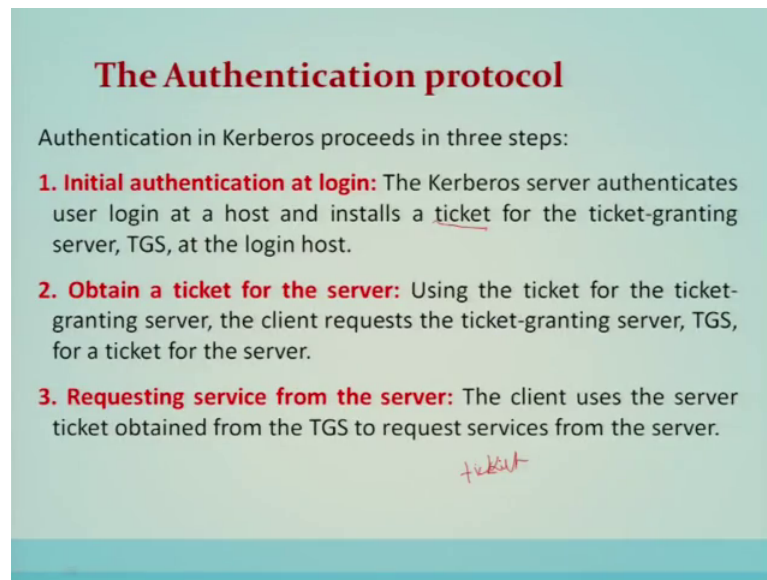


The diagram shows a client box labeled 'client (U, password)' with an arrow pointing to a larger box labeled 'KS' (Kerberos Server). Above the KS box is a vertical rectangle labeled 'Database'.

So, this is Kerberos server and the client has to register by providing its user id and the password. So, the Kerberos server computes a key for that particular user or for that client that is called ku by applying a hash function on a password and this is one way function and it stores this particular key in a database. So, it has a database constructed of all the registered clients on Kerberos authentication server and the passwords are is stored by hashing that is one big hash function is used and that password is stored in the Kerberos database for that user.

So, authentication in Kerberos proceeds in 3 different steps. So, initial authentication at the login.

(Refer Slide Time: 13:04)



The Authentication protocol

Authentication in Kerberos proceeds in three steps:

- 1. Initial authentication at login:** The Kerberos server authenticates user login at a host and installs a ticket for the ticket-granting server, TGS, at the login host.
- 2. Obtain a ticket for the server:** Using the ticket for the ticket-granting server, the client requests the ticket-granting server, TGS, for a ticket for the server.
- 3. Requesting service from the server:** The client uses the server ticket obtained from the TGS to request services from the server.

ticket

So, Kerberos server authenticates user login at the host and installs a ticket for the ticket granting server at the login host, the second step says that obtain a ticket for the server and third is requesting service from the server. So, the terminology which is coming at all 3 step is called a ticket. So, the Kerberos authentication protocol will not expose user password on the network rather it is a ticket based system. So, Kerberos authentication server will issue the ticket for a particular session to the client and client will show that ticket to the server and server can understand the authentication and client and server can communicate thereafter with the secret key which this shared in this particular protocol in these stages.

So, how the shared keys between the client and server is being exchanged with the help of a Kerberos server that we will see and what is the ticket why, how the ticket is basically solving the problem of not exposing the passwords on the network that also we are going to see. So, all 3 steps will involve the tickets. So, how the tickets are generated without exposing the password that we are going to see. So, the steps all 3 steps are shown over here. So, the components I explained you that Kerberos contains the Kerberos authentication server.

(Refer Slide Time: 14:49)

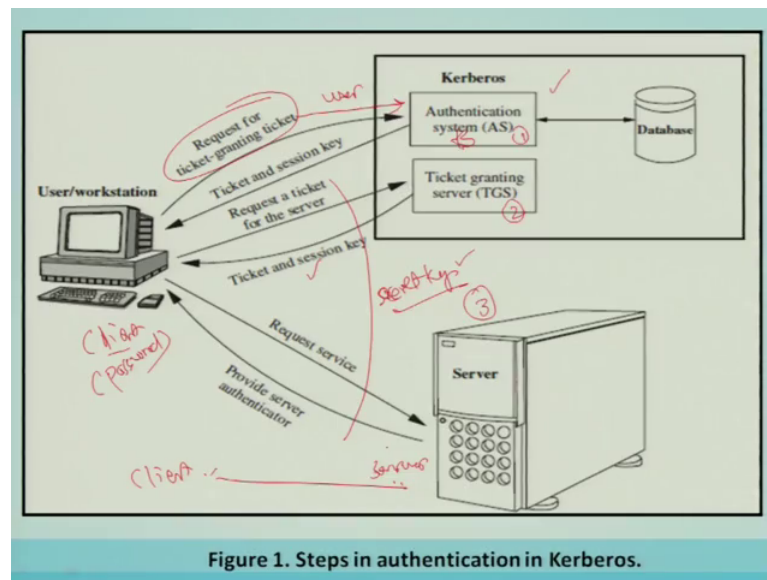


Figure 1. Steps in authentication in Kerberos.

Then second component here in this Kerberos authentication protocol is ticket granting server and third is the actual server where the client and server this is called client, we are the client want to communicate with the server.

So, this particular process will do the authentication, you see in nowhere the password is now flowing. So, user will send the request for ticket granting ticket with putting his username and authentication server will give the ticket and a session key and now then client will use its password there itself in the on the host to decrypt this particular ticket and session key because the client is already registered with the Kerberos with its user id and a password. So, now, the password will not flow and then it will extract the ticket and session key and then make a request for this particular server to a ticket granting server.

So, ticket granting server through the Kerberos authenticator it will try to understand our identity of the client and then issue a ticket and a session key for this particular server. Client will use this ticket and the session key and directly get the request for the service from the server and the server will authenticate, and in this particular in these process a secret key will be exchanged between the client and the server and thereafter all the messages which client will send using that particular secret key it will be encrypted and send.

(Refer Slide Time: 17:16)

Step 1: Initial authentication at login

- Initial authentication at login uses a Kerberos server and is shown in **Algorithm 1**. Let **U** be a user who is attempting to log into a host **H**.

(1)	$U \rightarrow H$:	U
(2)	$H \rightarrow \text{Kerberos}$:	U, TGS
(3)	Kerberos	:	retrieve k_U and k_{TGS} from database generate new session key k create a ticket-granting ticket $tick_{TGS} = \{U, TGS, k, T, L\}_{k_{TGS}}$
(4)	Kerberos $\rightarrow H$:	$\{TGS, k, T, L, tick_{TGS}\}_{k_U}$
(5)	$H \rightarrow U$:	"Password?"
(6)	$U \rightarrow H$:	password
(7)	H	:	compute $k'_U = f(\text{password})$ recover $k, tick_{TGS}$ by decrypting $\{TGS, k, T, L, tick_{TGS}\}_{k_U}$ with k'_U if decryption fails, abort login, otherwise, retain $tick_{TGS}$ and k erase password from the memory

Algorithm1: Initial authentication at login

So, the information which will be flowing on the network will be secured, having the secure communication. So, I have explained you the real application and the use of Kerberos authentication process let us see all three steps of the Kerberos authentication.

So, initial authentication at the login uses at a Kerberos server and is shown in the algorithm this is the algorithm, let u be a user who is attempting to log on to a host. So, user will give its identity not in the form of a password, but only, but it has to identify the name of the user to the host. Let us say it is U who want to use the Kerberos authentication service or who want to use the server, but it has to a authenticated with the help of Kerberos authentication protocol. So, this particular host so user and host there are two things means in the normal system user can come put its login and can start accessing to the host.

Now, in the distributed system it is connected to the network of several other things. So, user has to give its identity to the host and host will send this identity user for a ticket granting server service to the Kerberos. Kerberos will retrieve the k u and k of TGS from the database based on the username record which is stored in its database in the Kerberos and then generate a new session key k and create a ticket granting ticket and this particular ticket that is that will be generated by the Kerberos is in this particular form which is encrypted with ticket granting service.

That is a secret key of a ticket granting service and this is given to the host, host after receiving this particular ticket, it will since it is encrypted with the secret password of U. So, as far as the user is concerned it will produce its password and use the password to decrypt this particular message which is sent by the Kerberos, recover a session key, recover the ticket for ticket granting service and if the decryption fail then the login will also be failed.

So, this is the initial authentication at the login step.

(Refer Slide Time: 20:17)

Contd...

- In step 1, user U initiates login by entering his/her username. ✓
- In step 2, the login host H forwards the login request and the i.d. of the TGS to a Kerberos server. ✓
- In step 3, the Kerberos server retrieves k_u and k_{TGS} from the database, generates a new session key k and creates a ticket-granting ticket $ticket_{TGS} = \{U, TGS, k, T, L\}_{k_{TGS}}$, where U is the identity of the user who wishes to communicate with the server, TGS is the identity of the ticketgranting server, k is the session key, T is a timestamp, L is the ticket's lifetime and k_{TGS} is the key shared between the TGS and the Kerberos server. ✓

(network message doesn't flow password)

\square_{KS}
 \square_{TGS}

So, it is explained again here in this particular slide. So, in a step one user u initiates login by entering his username here there is no password is given. In step number 2 log in host forwards the login request and the id of the ticket granting service to the Kerberos. So, there no password is being sent only the login request and the id of ticket granting service is given to the Kerberos you send on the network. So, network message does not close the password so that you have to observe.

(Refer Slide Time: 21:06)

Contd...

- In step 4, the Kerberos server encrypts the ticket **tickTGS**, the identity of the **TGS**, the session key, the timestamp, and lifetime with **ku** and sends it to **host H**.
- In step 5, on receiving this message from the Kerberos server, host H prompts the user for his/her password, which the user supplies in step 6. In step 7, host H computes the key, **K'u**, corresponding to the password using the **one-way function f**. ✓
- The host recovers the session key **k** by decrypting **{TGS, k, T, L, tickTGS}ku** with **k'u**. ✓
- If the password supplied by the user is not the valid password of **U**, **k'u** would not be identical to **ku**, and the authentication will fail.

So, in step number 3 Kerberos server retrieves the secret key of U and also the key of a ticket granting service from the database generating a new session key **k** and creates a ticket granting ticket as I explained you. We use the identity of the user who wishes to communicate TGS is the identity of the ticket granting server and **K** is the session key **T** is the timestamp **L** is the tickets lifetime **K** of TGS is the key shared between TGS and Kerberos. So, this is the secret key between the Kerberos and the ticket granting service. So, the ticket granting service this particular message is encrypted with **KTGS**. So, then step number 4 Kerberos server encrypts the ticket granting service the identity of TGS the session key and so on gives to the host.

The step number 5 on receiving this message from Kerberos server host **h** prompts the user for his password and using that particular password it will decrypt this particular message which is sent by the Kerberos and recovers the session key **K** which it is going to use it. So, the so, session key will be retrieved out of this particular message thus the user is authenticated if the host is able to decrypt the message which is received by the Kerberos, upon successful authentication the host saves the new session key **K** and a ticket granting ticket for further use and erases the password from the memory.

(Refer Slide Time: 22:55)

Contd...

- Thus, the user is authenticated if the host is able to decrypt the message for the Kerberos server.
- Upon successful authentication, the host saves the new session key k and the ticket-granting ticket, $tick_{TGS}$, for further use and erases the user password from the memory.
- The ticket-granting ticket is used to request server tickets from the TGS. Note that $tick_{TGS}$ is encrypted with k_{TGS} , the key shared between the TGS and the Kerberos server.

So, password use is over and the key and ticket granting, ticket for the ticket granting service both are restored, the ticket granting ticket is used to request the server tickets from the ticket granting service note that the tickets are encrypted using K of TGS. So, the key shared between TGS and server Kerberos server. So, the user will not be able to decrypt this part so this part will go as it is without decryption.

(Refer Slide Time: 23:37)

Step 2: Obtain a ticket for the server

- The client executes the steps shown in **Algorithm 2** to request a ticket for the server from the TGS. Basically, the client sends the ticket $tick_{TGS}$ to the TGS, requesting a ticket for the server S . (T_1 and T_2 are timestamps.)

(1)	$C \rightarrow TGS :$	$S, tick_{TGS}, \{C, T_1\}_k$
(2)	$TGS :$	recover k from $tick_{TGS}$ by decrypting with k_{TGS} , recover T_1 from $\{C, T_1\}_k$ by decrypting with k , check timelines of T_1 with respect to local clock generate new session key k Create server ticket $tick_s = \{C, S, k, T, L\}_{k_s}$
(3)	$TGS \rightarrow C :$	$\{S, k, T, L, tick_s\}_D$
(4)	$C :$	recover $k, tick_s$ by decrypting the message with k

Algorithm 2: Obtain a ticket for the server

So, this is how the client will now communicate with the ticket granting service. So, as I told you that this ticket of TGS which is encrypted will go as it is. So, the client executes

the I step shown in the algorithm to request the ticket for the server from TGS basically the client sends the ticket TGS to TGS requesting a ticket for the server s. So, ticket granting service will issue that ticket for that particular server session. So, the first step is the client will request to the ticket granting service through this particular message some part of the message is basically received by the Kerberos authentication service. So, that ticket granting service will authenticate that this is the client which is authorized by the Kerberos server and the request for that server name is being encrypted and sent.

So, ticket granting service after receiving this particular message recovers K, K means the session key from TGS by decrypting with that particular key the secret key between Kerberos and the ticket granting service. Then it will recover the T1, T1 is the time stamp from C of T which is decrypted with that session key to check the time lines of T1 if the. So, if the lifetime of this particular ticket is over then it will not be allowed to make a session that is why. So, why this is done is to avoid the replay attack, with respect to the local clock and it will generate a new session key K. So, create the server ticket TS which is encrypted with that particular session key. So, it will contain the new session key the client credentials and the server also credentials and the timestamp and the lifetime of this particular ticket, this particular message TGS will send to C encrypted with the session key.

Now, C will recover k this k and this particular ticks, ticks is basically the ticket for the server which is sent by the ticket granting service by decrypting the message with its own key that is K. So, because the ticket is susceptible to the interception and replay it does not by itself constitute sufficient proof of identity.

(Refer Slide Time: 26:43)

Contd...

- Because a ticket is susceptible to interception and replay, it does not by itself constitute sufficient proof of identity.
- For authentication, a principal presenting a ticket must also demonstrate the knowledge of the session key k named in the ticket. An authenticator, $\{C, T\}_k$, where **C is the client identity**, T is the timestamp, and k is the session key, provides the demonstration.

+ timestamp

For authentication the principle presenting a ticket must also demonstrate the knowledge of session key K named in the ticket. So, the authenticator which contains the timestamp to avoid this replay attack and also this is the client's identity this is called authenticator, this also will be given or produced to the ticket granting service.

So, K is the session key provides this particular demonstration.

(Refer Slide Time: 27:35)

Contd...

- In **step 1**, to request a ticket for server S , client C presents its ticket-granting ticket $ticket_{TGS}$ along with the authenticator to the TGS. C 's knowledge of k is demonstrated using the authenticator $\{C, T\}_k$.
- In **step 2**, the TGS decrypts $ticket_{TGS}$ with key_{TGS} to recover k , verifies the authenticity of the authenticator by decrypting $C\|k$ with k , and checks the timeliness of T_1 in the authenticator and T in $ticket_{TGS}$.
- If both decryptions in step 2 are successful and T_1 is timely, the TGS is convinced of the authenticity of the ticket, and creates a ticket $ticks = \{C, S, k, T, L\}_{ks}$ for server S , where C is the identity of the client, S is the server identity, k is the new session key, T is the timestamp of the TGS, L is the lifetime of the ticket, and ks is the key shared between the TGS and server S .
- This ticket is returned to C in step 3. In step 4, C recovers k and ticks from $\{S, k, T, L, ticks\}_{ks}$ by decrypting it with k .

So, in a step number 1 to request a ticket for the server the client C presents its ticket granting ticket along with the authenticator that I explained. In a step number 2 ticket

granting service decrypts the ticket, ticket granting ticket with its secret key which is being encrypted with the help with by the Kerberos and we shared with that particular ticket granting service and this will recover the expression key K will verify the authenticity of the authenticator by decrypting the authenticator that is the identity of the client and the timestamp T1. And this is has to be decrypted with the with the key K, if both the decryptions in step number 2 are successful and T1 is timely 30 ticket granting service is convinced of the authenticity of the ticket and create an a ticket for the server. This ticket is returned to the client in step number 3 and in step number four the client recovers K session key and ticks that is the ticket for the server by decrypting it with K.

Third step is now the client will request the service from the server using this much of information which it has obtained by communicating with Kerberos and communicating with ticket granting service.

(Refer Slide Time: 29:11)

Step 3: Requesting service from the server

- Client C sends the ticket and the authenticator to server. The server decrypts ticks and recovers k. It then uses k to decrypt the authenticator $\{C, T_2\}_k$ and checks if the timestamp is current and the client identifier matches with that in the ticks before granting service to the client. If mutual authentication is required, the server returns an authenticator (**Algorithm 3**).

(1) $C \rightarrow S: tick_s, \{C, T_2\}_k$

(2) $S: recover k from tick_s by decrypting it with $k_s$$
 $: recover T_2 from $\{C, T_2\}_k$ by decrypting with $k$$
 $\checkmark \checkmark: check timeliness of T_2 with respect to the local clock$

(3) $S \rightarrow C: \{T_2+1\}_k$

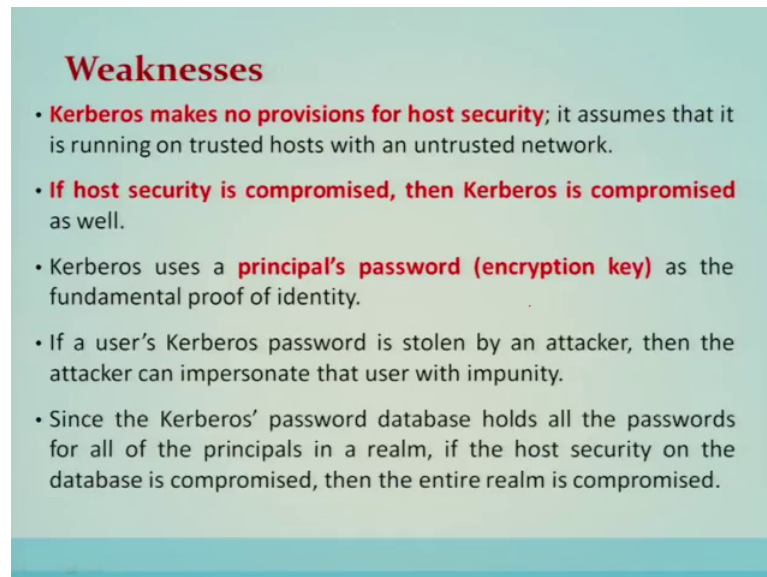
Algorithm 3 Requesting service from the server.

So, ticket granting service has given ticket to communicate with the server and also it contains a session key or a secret key which will be used for encryption between client and the server, let us see these steps. So, the client will send to the server that encrypted ticket and also the authenticator of the client and the timestamp which will contain and which will have the session key which is given by the ticket granting server.

Now, S the server will recover the session key K from ticks by decrypting it with its secret key Ks it will recover t two by decrypting with K and check the timeliness of that

time is time T2 with respect to the local clock. Now server will communicate to the client with the new timestamp T2 plus 1 and which is encrypted with the, with the common key K which is being given by the ticket granting service.

(Refer Slide Time: 30:35)



Weaknesses

- **Kerberos makes no provisions for host security**; it assumes that it is running on trusted hosts with an untrusted network.
- **If host security is compromised, then Kerberos is compromised** as well.
- 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 Kerberos' password database holds all the passwords for all of the principals in a realm, if the host security on the database is compromised, then the entire realm is compromised.

So, all these things are explained. So, the weaknesses is that there is no provision for the host security; that means, the passwords are stored in the host and if the host is untrusted then it will become a problem to the entire network.

So, this problem is resolved in the Kerberos version 5 which will introduced the pre authentication to solve this particular problem that is explained here in the slide. Now the next important protocol is secure socket layer protocol. So, the secure socket layer protocol was developed by Netscape and is a standard internet protocol for secure communication. So, the Kerberos authentication was that without flowing the password on the inter on the network how the authentication is to be carried out with the help of servers.

(Refer Slide Time: 30:54)

Contd...

- In **Kerberos version 4**, authenticators are valid for a particular time. If an attacker sniffs the network for authenticators, they have a small time window in which they can re-use it and gain access to the same service.
- **Kerberos version 5 introduced a replay cache** that prevents any authenticator from being used more than once.
- Since anybody can request a ticket-granting ticket 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, say using the dictionary attack.
- **Kerberos version 5 introduced pre-authentication** to solve this problem.

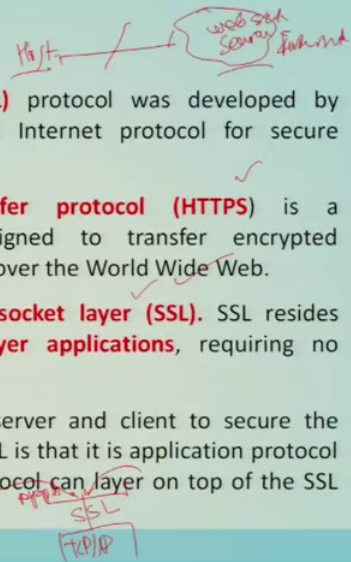
Now another protocol called SSL where in the host if it want to access to a website or a server on the internet then what are the problems it is going to face is that authentication, how it is going to carry out the authentication.

So, we are going to use this particular protocol for the authentication on the internet

(Refer Slide Time: 32:10)

SSL Protocol

- The **secure sockets layer (SSL)** protocol was developed by **Netscape** and is the standard Internet protocol for secure communications.
- The **secure hypertext transfer protocol (HTTPS)** is a communications protocol designed to transfer encrypted information between computers over the World Wide Web.
- **HTTPS** is http using a **secure socket layer (SSL)**. SSL resides between **TCP/IP and upper-layer applications**, requiring no changes to the application layer.
- SSL is used typically between server and client to secure the connection. One advantage of SSL is that it is application protocol independent. A higher-level protocol can layer on top of the SSL protocol transparently.



And if the authentication between the hosts in the website; that means, host how it has to authenticate then it is communicating with the proper website and the proper website has to authenticate to the to the host this is very important in the sense. If let us say it is a

payment system or a banking payment system then the authentication of the host as well as the website is very very important otherwise credit card information may be had or may be leaked and it will siphon out all the money. So, this particular SSL protocol is going to be very useful, we are going to see the use of this SSL protocol.

The secure hypertext transfer protocol HTTPS is basically a communication protocol designed to transfer encrypted information between computers and worldwide web. HTTPS is http using secure socket layer SSL, SSL resides between the TCP IP and the application, requiring no changes to the application layer the different application can use this SSL to get this particular communication secure. So, SSL is used typically between server and the client to secure the connection, one advantage of SSL is that it is an application independent protocol. So, higher level protocol can layer on top of SSL protocol transparently.

SSL protocol let us go and detail about the features of SSL protocol.

(Refer Slide Time: 33:55)

Features

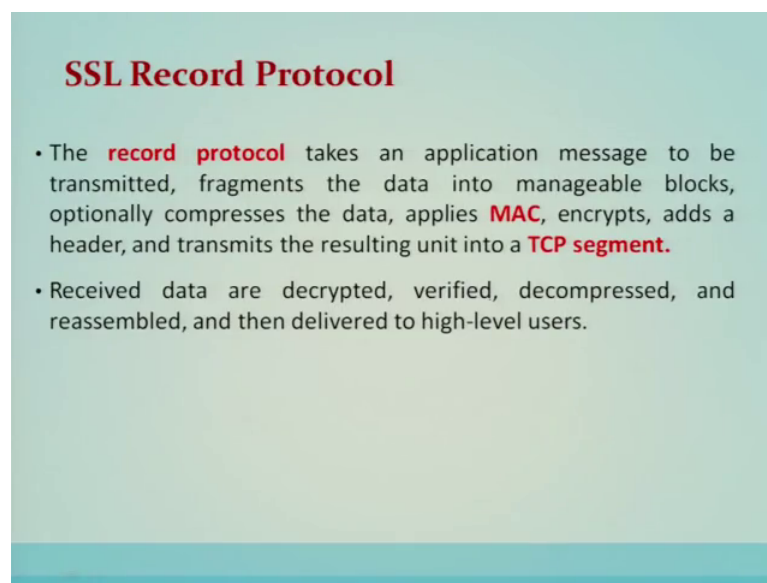
- SSL protocol allows client–server applications to communicate in a way so that eavesdropping, tampering, and message forgery are prevented.
- The SSL protocol, in general, provides the following features:
 - **End point authentication:** The server is the “real” party that a client wants to talk to, not someone faking the identity.
 - **Message integrity:** If the data exchanged with the server has been modified along the way, it can be easily detected.
 - **Confidentiality:** Data is encrypted. A hacker cannot read your information by simply looking at the packets on the network.

SSL protocol allows client server applications to communicate in a way. So, that each dropping, tampering, message forgery are prevented which is very important for the customers who are doing the transaction the banking related transactions where the information of credit card or a bank details are basically sent. So, SSL protocol in general provides the following features endpoint authentication the server is the real party that the client want to talk to, not someone faking the identity for example, if

whether it is a proper bank website or some other fake website so it has to be authenticated by the person or a client who want to use it. Then second part second feature of SSL is the message integrity if the data exchanged with the server has been modified along the way it can easily be directed. So, the messages which are being exchanged at both the principal has proper (Refer Time: 35:08).

Third one is confidentiality. So, the data which is flowing is encrypted hacker cannot read this information by simply looking at the packet on the network.

(Refer Slide Time: 35:29)



SSL Record Protocol

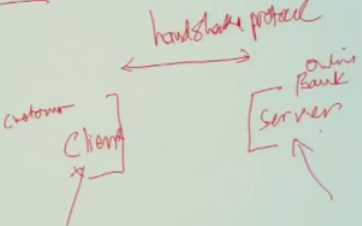
- The **record protocol** takes an application message to be transmitted, fragments the data into manageable blocks, optionally compresses the data, applies **MAC**, encrypts, adds a header, and transmits the resulting unit into a **TCP segment**.
- Received data are decrypted, verified, decompressed, and reassembled, and then delivered to high-level users.

So, SSL record protocol takes the application message and fragments the data into the manageable blocks, encrypts, add headers and the resulting unit is sent to the TCP.

(Refer Slide Time: 35:55)

SSL Handshake Protocol

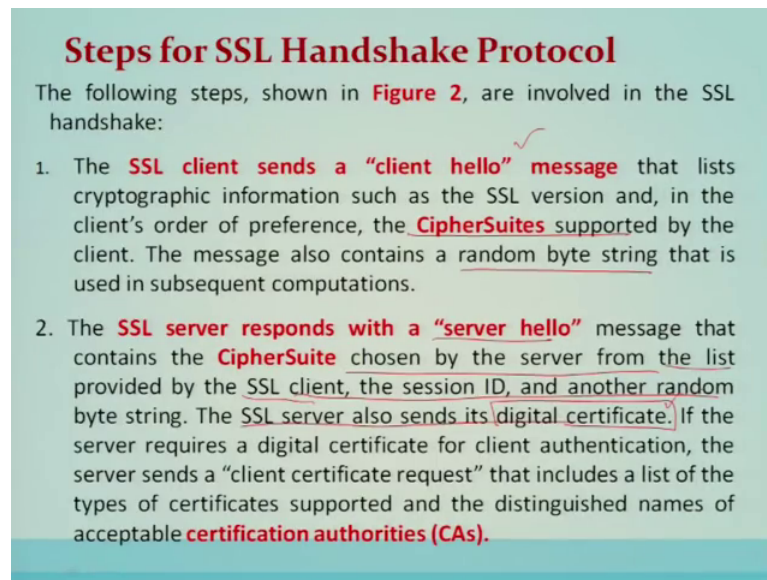
- The **SSL handshake protocol** allows the server and client to authenticate each other and to negotiate an encryption algorithm and cryptographic keys before the application protocol transmits or receives its first byte of data.



The diagram illustrates the SSL Handshake Protocol. It shows two entities: a 'Customer' (labeled as 'Client') and an 'Online Bank' (labeled as 'Server'). A double-headed arrow between them is labeled 'handshake protocol', indicating a bidirectional communication process. The 'Customer' is represented by a box with a checkmark, and the 'Online Bank' is represented by a box with an arrow pointing towards it.

Now, another important part of this SSL protocol is the handshake protocol. So, the client and the server so if the client is let us say a customer of a bank and the server is let us say online bank then the SSL protocol first it will handshake and establish the parameters on which they are going to provide you the secure way of communication, that is why it is called handshake protocol. So, allows the server and the client to authenticate each other and to negotiate an encryption algorithm and cryptographic keys before application transmits or receives its first byte of data, this is very very important. That means, both they will communicate and negotiate this information; that means, which encryption algorithm it is going to use and if it is decided then what is the keys and they have to exchange once this is being negotiated and also it is supported at the clients browser and at the servers website. So, those technical details are negotiated and they will follow this in further communication for the secure way of communication.

(Refer Slide Time: 37:32)



Steps for SSL Handshake Protocol

The following steps, shown in **Figure 2**, are involved in the SSL handshake:

1. The **SSL client sends a "client hello" message** that lists cryptographic information such as the SSL version and, in the client's order of preference, the **CipherSuites supported** by the client. The message also contains a **random byte string** that is used in subsequent computations.
2. The **SSL server responds with a "server hello" message** that contains the **CipherSuite** chosen by the server from the list provided by the **SSL client, the session ID, and another random byte string**. The **SSL server also sends its digital certificate**. If the server requires a digital certificate for client authentication, the server sends a "client certificate request" that includes a list of the types of certificates supported and the distinguished names of acceptable **certification authorities (CAs)**.

Let us see the handshake protocol of SSL. So, first SSL client sends a client hello message and in this hello message it will give the information about the clients order of preferences, about which of these encryption algorithms it is supporting at its end and also some random byte signature is being sent on this particular hello message. SSL server it will respond to the hello message after choosing from the options given by the client about the encryption methods, encryption algorithms or the cryptographic keys from this particular list it will it will choose and which is provided by the client. So, so SSL server also sends digital certificate. So, digital certificate is basically a certificate by which the client can authenticate that this is the correct website.

So, digital certificate is a cert is provided through a certification agencies, which is well known to everyone for example, if let us say a certificate is taken out from some company let us say x which is not well known then that certificate is not authorized certificate or is not known to be a proper certificate. Now, if Microsoft gives a certificate Microsoft is well known company then that certificate is authenticated or properly verified. So, digital serve certificates are issued from certification agencies which are well known. So, if a server requires the distance certificate from the client also for authentication then server also informed that please send the client certificate, that includes the list of types of certificate supported and the distinguished names of acceptable certification authorities, also it will server will provide that from these listed certification authority certificate has to be provided by the client.

(Refer Slide Time: 39:44)

Contd...

3. The **SSL client verifies the digital signature** on the SSL server's digital certificate and checks that the **CipherSuite** chosen by the server is acceptable.
4. The SSL client, using all data generated in the handshake so far, **create a premaster secret for the session** that enables both the client and the server to compute the secret key to be used for encrypting subsequent message data. The premaster secret itself is encrypted with the server's public key.
5. **If the SSL server sent a "client certificate request,"** the SSL client sends another signed piece of data which is unique to this handshake and known only to the client and server, along with the encrypted premaster secret and the client's digital certificate, or a "no digital certificate alert." This alert is only a warning, but with some implementations the handshake fails if client authentication is mandatory.

Now, SSL client verifies the digital signature on SSL server's digital certificate and checks the cipher suite chosen by the server is acceptable, SSL client using all the data generated in the handshake so far create a pre master secret for the session. If SSL server sent a client certificate request the SSL client sends another signed piece of data which is unique to this and will obtain the digital certificate and send it in the part of handshake.

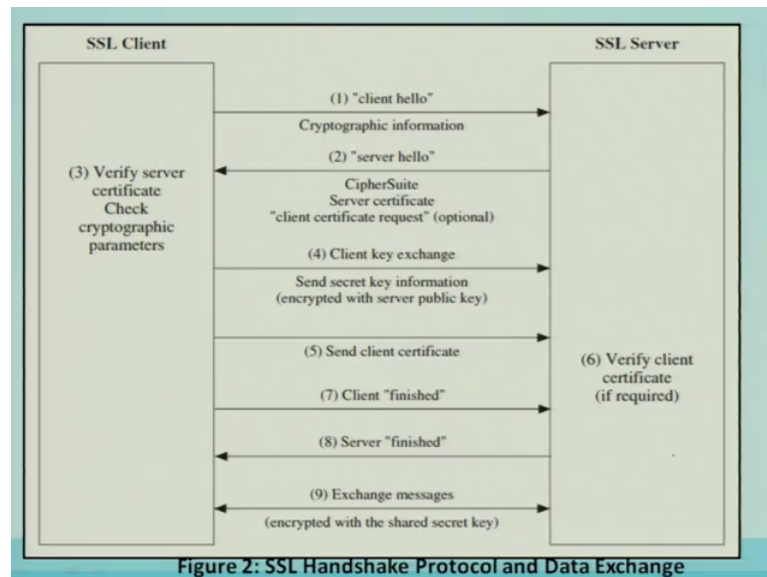
(Refer Slide Time: 40:25)

Contd...

6. The **SSL server verifies the signature** on the client certificate.
7. The **SSL client sends the SSL server a "finished" message**, which is encrypted with the secret key, indicating that the client part of the handshake is complete.
8. The **SSL server sends the SSL client a "finished" message**, which is encrypted with the secret key, indicating that the server part of the handshake is complete.
9. For the duration of the **SSL session**, the SSL server and SSL client can now exchange messages that are encrypted with the shared symmetric secret key.

So, SSL server verifies that such a signature on the client certificate, SSL client sends SSL server a finished message; SSL server sends SSL client a finished message on the other side for the duration of SSL session.

(Refer Slide Time: 40:54)



SSL server and SSL client can now exchange the messages that are encrypted with the shared symmetric key which are exchanged in this handshake.

So, all the handshake a steps which I have told that is being listed over here in this handshake protocol. So, both client and server authentication there is a step that requires data to be encrypted with one of the keys in the asymmetric key pair and is decrypted with the other key of the pair.

(Refer Slide Time: 41:25)

How SSL provides authentication

- **During both client and server authentication**, there is a step that requires data to be encrypted with one of the keys in an asymmetric key pair and is decrypted with the other key of the pair .
- **For server authentication**, the client uses the server's public key to encrypt the data that is used to compute the secret key. The server can generate the secret key only if it can decrypt that data with the correct private key.
- **For client authentication**, the server uses the public key in the client certificate to decrypt the data the client sends during step 5 of the handshake. The exchange of finished messages that are encrypted with the secret key (steps 7 and 8 in the overview) confirms that authentication is complete.

For server authentication the client uses servers public key to encrypt the data that is used to compute the secret key the server can generate the secret key only if it can decrypt that particular data with the correct private key. So, for that basically will do the authentication of a server so only genuine server or authenticated server can only be able to do this step. For client authentication the server uses the public key in the client certificate to decrypt the data the client sends during step number 5 for handshake, the exchange of finished information messages that are encrypted with the secret key of the of the client confirms that that authentication is complete.

(Refer Slide Time: 42:24)

Contd...

- **The SSL client needs:**
 - The CA certificate for CA Y or the personal certificate issued to the server by CA Y .
- If the SSL server requires client authentication, the server verifies the client's identity by verifying the client's digital certificate with the public key for the CA that issued the personal certificate to the client, in this case CA X. For both server and client authentication, the SSL server needs:
 - The personal certificate issued to the server by CA Y .
 - The server's private key.
 - The CA certificate for CA X or the personal certificate issued to the client by CA X.

So, SSL server requires the client authentication, the server verifies the clients identity by verifying the clients digital certificate that I have already explained. So, conclusion authentication is a process by which one principle verifies the identity of the other principle.

(Refer Slide Time: 42:51)

Conclusion

- **Authentication is a process by which one principal verifies the identity of other principal.** For example, in a client–server system, the client and the server may need to verify each other’s identity to assure that each is talking to the right entity. *SSL*
- Generally, authentication is based on the possession of a secret information, like password, that is known only to the entities participating in the authentication. For a successful authentication, the entity must demonstrate the knowledge of the right secret information. *password - network*
- In this chapter, we described the authentication protocols such as **Kerberos and Secure Socket Layer (SSL) protocol.**

So, it involves the identification is to be produced, identity is to be produced in the form of digital certificates and these particular and also with the help of secret key in a public in a asymmetric key cryptosystem.

So, for example, in a client server system the client and server may need to verify each other’s identity to assure that each is talking to the right entity that we have seen, in both these protocols that is in the Kerberos as well as in the SSL. Generally authentication is based on possession of the secret information like password that is known only to the entities participating in the authentication, for a successful authentication the entity must demonstrate the knowledge of the right secret information.

So, most importantly we have seen 2 different type of protocol one is Kerberos wherein the password is protected from the network and the other protocol SSL wherein how the secure communication between the two partners that is client and server can takes place on the internet. So, both this particular protocol has lot of use and is being heavily used and we have basically able to express it in this part of the lecture the details.

Thank you.