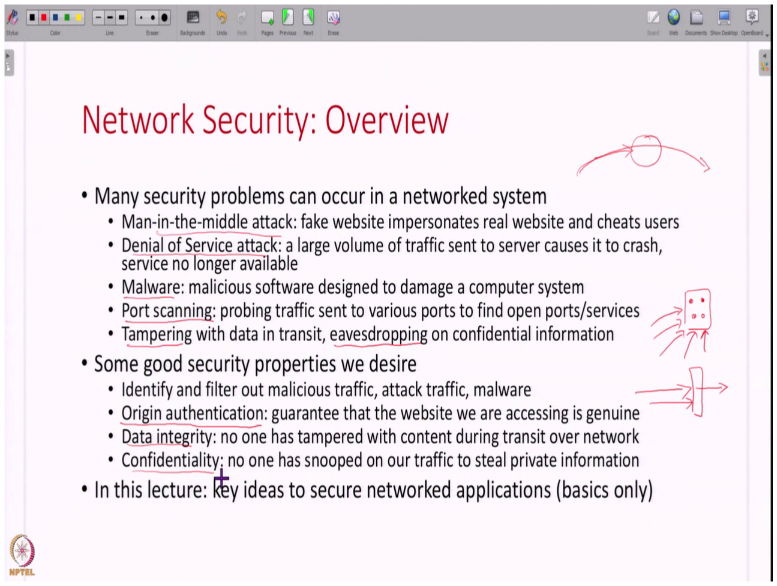
Design Engineering of Computer Systems
Professor. Mythili Vutukuru
Computer Science and Engineering
Indian Institute of Technology, Bombay
Lecture 35
Network Security

Hello, everyone. Welcome to the twenty fifth lecture in the course Design and Engineering of Computer Systems. So, in this week, we have studied briefly how computer networks work. So we have not gone into a lot of detail because that is a full course by itself. But we have studied a few important concepts that will help us when we go towards this end to end design of computer systems.

So, in this lecture, what we are going to do is, similarly, we are going to briefly touch upon some important concepts in network security. Again, a full-fledged treatment of, this is a full course by itself, but in this one lecture, we will just briefly touch upon some very important concepts. So, let us get started.

So, what is network security? So, whenever we access any Internet service over the network, a lot of bad things can happen. And network security is sort of just the mechanisms that are in place to prevent or to at least reduce the impact of these bad things. So, what are some of the security problems that you can face in any computer system?

(Refer Slide Time: 01:22)



Network Security: Overview

- Many security problems can occur in a networked system
  - Man-in-the-middle attack: fake website impersonates real website and cheats users
  - Denial of Service attack: a large volume of traffic sent to server causes it to crash, service no longer available
  - Malware: malicious software designed to damage a computer system
  - Port scanning: probing traffic sent to various ports to find open ports/services
  - Tampering with data in transit, eavesdropping on confidential information
- Some good security properties we desire
  - Identify and filter out malicious traffic, attack traffic, malware
  - Origin authentication: guarantee that the website we are accessing is genuine
  - Data integrity: no one has tampered with content during transit over network
  - Confidentiality: no one has snooped on our traffic to steal private information
- In this lecture: key ideas to secure networked applications (basics only)

So you can have various things go wrong. For example, you can have what is called the man in the middle attack. That is, you think you are accessing some websites, say your bank's website, but instead of going to that website, somebody in the middle pretends to be the bank, takes all your account details, and can steal some sensitive information from you, and so on. So that is the man in the middle attack.

That is somebody impersonating somebody else on the Internet. You can also have a denial of service attack. That is some malicious attackers want to bring down a website or a web service. So they send a large volume of traffic to the server and cause the server to crash. So they cause the server to deny service to legitimate clients.

Then you have all sorts of malware on the Internet. You have viruses and so on. Malicious software that is somehow injected via the network into your computer system and it can damage different parts of the computer system. Then you have other more sort of specialized attacks like what is called port scanning.

On your server, you can have many different processes, an email server, web server, different processes listening on different port numbers. So what some attackers can do is they can keep sending SYN packets, opening a connection to many different ports at random on your computer to see which ports are open.

Then if they find some port is open, some server is running, then they will try to attack that server. You can just randomly scan ports, which is not something that regular clients would do. So, therefore, this is the beginning of an attack in some sense. And you can have all sorts of other things like when you are sending your data over the network, somebody can tamper with your data packets, or even if they do not tamper with them actively, they can do, tampering is sort of more active, or they can just passively eavesdrop on your data packets and learn some confidential information about the user or the computer system. So, all of these are various things that can go wrong in a computer network today.

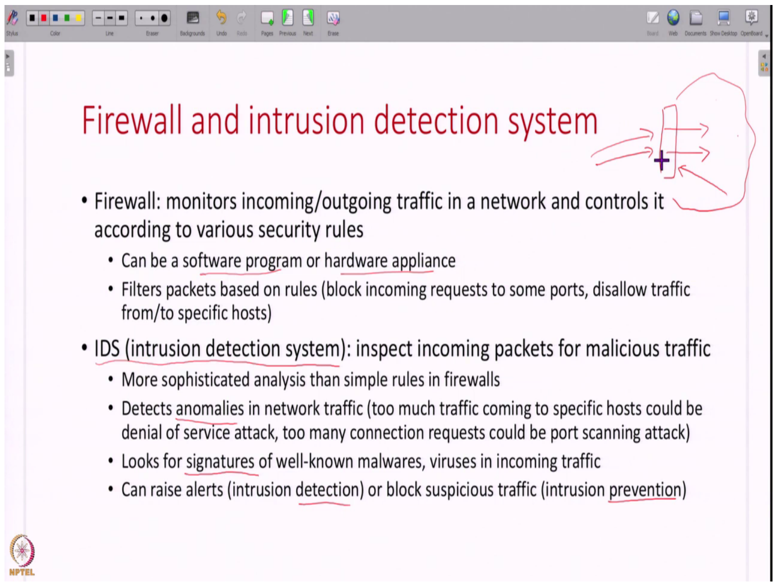
And of course, not a lot of bad things happen because we have mechanisms in place to protect us against all of these attacks. In this lecture, we are going to briefly understand what some of these mechanisms are. And some of the properties that we want from these mechanisms from the network security solutions are, we want a way to identify and stop all bad traffic, any attack traffic, malware traffic, any bad thing that is coming into your network, you kind of want to filter it out and send only the good traffic in. You need some mechanism for this.

Then on the Internet today, you need what is called origin authentication. That is when you are accessing a website, especially for trusted services like an online banking service or something you need to know that the website is actually genuine. It is actually who they say they are. You need some kind of authentication of the origin.

Then we need integrity of data. What does it mean? Whatever data you send, it reaches the other side without any tampering, or anybody modifying it and injecting some bad content into it in any way. Then we also need confidentiality. We do not want anybody to snoop on our traffic and steal any private information.

So, all of these terms, I would like you to understand origin authentication, data integrity, confidentiality, and these are all the properties that we will see how today's solutions provide all of these properties. And of course, there are many solutions in use today for all of these things. And this one lecture cannot cover all of them. But we will briefly cover just two or three key ideas in the area of Internet security.

(Refer Slide Time: 05:32)



Firewall and intrusion detection system

- Firewall: monitors incoming/outgoing traffic in a network and controls it according to various security rules
  - Can be a software program or hardware appliance
  - Filters packets based on rules (block incoming requests to some ports, disallow traffic from/to specific hosts)
- IDS (intrusion detection system): inspect incoming packets for malicious traffic
  - More sophisticated analysis than simple rules in firewalls
  - Detects anomalies in network traffic (too much traffic coming to specific hosts could be denial of service attack, too many connection requests could be port scanning attack)
  - Looks for signatures of well-known malwares, viruses in incoming traffic
  - Can raise alerts (intrusion detection) or block suspicious traffic (intrusion prevention)

So, the first thing we begin with are what are called firewalls or intrusion detection systems. So, what is a firewall? A firewall is a piece of, can either be a software program or hardware box that is usually put at the entry point to a network. All the traffic coming into your network it goes through this firewall either the software or hardware box, and then only the traffic that clears the firewall is permitted into any network. So, what does a firewall do?

It will filter packets based on some simple rules. For example, if you do not have any web servers, then you should not have any traffic coming into port 80 into your system. If it is coming in, maybe it is malicious traffic that is doing some port scanning. By using rules like that, for certain secure servers you do not want anybody apart from authenticated some specific users to handle, to access the secure servers.

You do not want traffic from hosts inside the network to go to certain specific websites or IP addresses. A firewall basically filters traffic that is both coming into your network and going out from your network based on some simple rules. And these rules, who sets these rules. These rules are configured by the network administrators. Whoever is managing this network will configure these rules in a firewall in order to block some ports or some traffic from some IP addresses, whatever.

There are various formats of these rules. And in that way a firewall ensures some amount of security. And in addition to firewalls large networks today also deploy what are called intrusion detection systems or IDSs. So, these IDSs do sort of one level deeper scrutiny than the simple rules of a firewall. They inspect all traffic to find various patterns. For example, they try to do some anomaly detection.

If at this time of the hour every day you only got an x amounts of traffic, but suddenly 10x traffic is coming to some specific host in your network, then that is an anomaly. It should not be happening. So, such anomalies, maybe that is a denial of service attack or too many SYN packets, connection requests are coming into a particular host to random port numbers that could be a port scanning attack. In this way, they will try to find some patterns, abnormal patterns, that is what anomalies are. So, these IDSs run some complicated algorithms to detect these abnormal patterns and maybe alert the sysad in the network or even block this anomalous traffic.

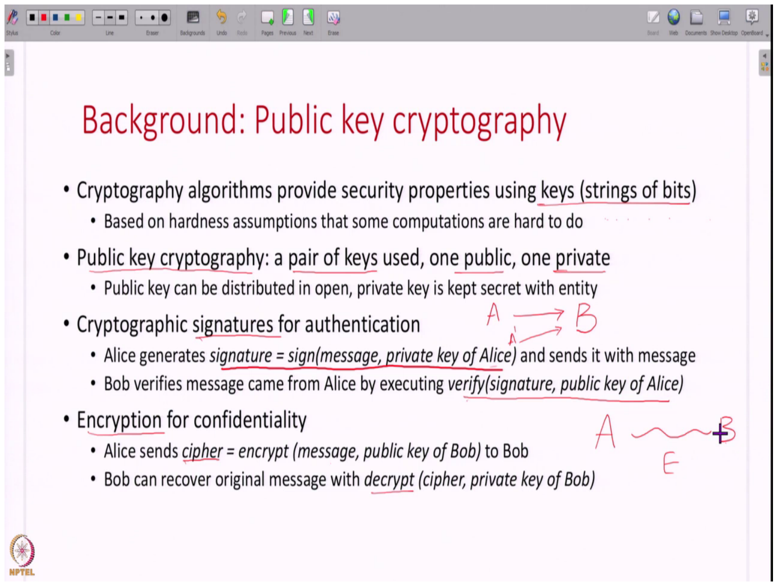
And the other thing they do is they look for signatures. Some viruses once they are well known that this virus is there, you can actually inspect the incoming traffic and search for the signature in the traffic. So, these IDS is also look for signatures of already known malware and viruses.

And they can raise alerts or block the traffic. This is raising an alert is called intrusion detection. If you block the traffic, that is called intrusion prevention.

You are kind of taking a more active role here. So, in this way, firewalls and IDSs are deployed at the entry point of most large networks today in order to stop malicious traffic. And of course, that does not mean all attack traffic is stopped. There are still clever attackers that can subvert these firewalls and intrusion detection systems, which is why you occasionally have these security attacks happening. But for the most part, these systems try to block a lot of the attack traffic.

So, the next idea we are going to study is we are going to understand how HTTP security, secure HTTP works on the Internet today. And before we go there, we need to get some background about what is called a field of study that is called cryptography. So, a full treatment of this is again a complete course. But in a couple of slides, I am going to briefly introduced you what is cryptography. And with that limited knowledge, we are going to understand how HTTPS or secure HTTP works. So, cryptography is of two types either public key or symmetric key.

(Refer Slide Time: 09:46)



Background: Public key cryptography

- Cryptography algorithms provide security properties using keys (strings of bits)
  - Based on hardness assumptions that some computations are hard to do
- Public key cryptography: a pair of keys used, one public, one private
  - Public key can be distributed in open, private key is kept secret with entity
- Cryptographic signatures for authentication
  - Alice generates *signature = sign(message, private key of Alice)* and sends it with message
  - Bob verifies message came from Alice by executing *verify(signature, public key of Alice)*
- Encryption for confidentiality
  - Alice sends *cipher = encrypt (message, public key of Bob)* to Bob
  - Bob can recover original message with *decrypt (cipher, private key of Bob)*

So, let us see a background of what is public key cryptography. So cryptography in general is basically a set of algorithms that provide some security properties. And at the base, at the core of these algorithms are what are called keys or strings of bits. So, you use some large array of bits. And using these keys, you run various computations on these keys. And you somehow guarantee some security properties. And there is a lot of theoretical underpinnings to cryptography. There are certain hardness assumptions saying some computations are hard to do.

So, of course, we would not go into all of these details here, but just know that people have, some very smart people have designed some very smart algorithms to provide certain security properties. And the proofs of why these algorithms work is beyond the scope of this course. So, one type of cryptography is what is called public key cryptography. In this cryptography, between the various entities that are doing some computations and guaranteeing security, between all of these entities, two keys, a pair of keys are used in all of these algorithms. One key is public is well known to everyone and the other key is private.

So, when you use two keys, a public key and a private key, that is called public key cryptography. And this public key can be distributed out in the open to anyone, whereas the private key is usually kept secret. And the secrecy of this private key is important for the security. So, what is public key cryptography used for?

The two common uses are one is what are called cryptographic signatures for authentication. So, that is suppose you have two people, Alice and Bob. These are usually the common names used in cryptography. You have two people Alice and Bob exchanging a message. Alice is sending a message to Bob. Now, when Bob gets this message, how does Bob know that this message came from Alice only and not from some other random person? We use signatures for this authentication.

So, Alice uses a signature to authenticate herself to Bob. That is Alice, basically, if there is a message, Alice will sign this message using its private key. Now, this private key is known only to Alice. Therefore, only Alice can generate this signature, and it will send the signature along with the message. And when Bob gets this message, Bob can run another cryptographic algorithm to verify the signature using the public key of Alice. So, Bob knows the public key of Alice.

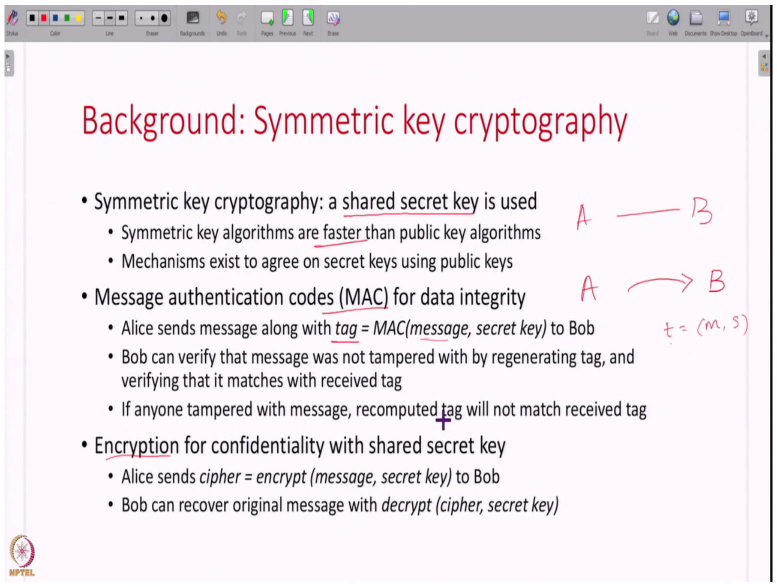
And with that public key and the signature, Bob will run some algorithm. And that algorithm will tell you if the signature was actually generated by whoever held the private key of Alice. Anybody else, any random person, any attacker say A prime cannot generate the signature. That will pass this verification test that Bob is running with the public key of Alice. Only the person that had the private key of Alice could have generated the signature.

In that way, there is some amount of authentication that this message actually came from Alice. Of course, it is not authenticated to Alice, the person, but it is authenticated, to be more specific, to the private key of Alice. Whoever held the private key of Alice has generated this message that can be verified by Bob using this signature.

So, the other use for public key cryptography is an encryption, which is used for confidentiality. So, suppose again Alice and Bob are exchanging a message, but they do not want anybody along the path. Alice is sending to Bob and any eavesdropper along the path should not be able to understand what is in this message. So, the way this is done is Alice instead of sending the message, Alice will encrypt the message using the public key of Bob and send only this cipher, the cipher text or the secret text will be sent to Bob.

Now, Bob, using his private key, can decrypt this message and recover the original message that Alice wanted to send. So, Alice knows the public key of Bob, whoever Alice wants to talk secretly to, Alice will know the public key, and encrypt, and Bob will decrypt the message. So, in this way, with public key cryptography you can do, you can ensure some kind of confidentiality, you can ensure some kind of authentication. And there are many widely used systems today that depend on public key cryptography.

(Refer Slide Time: 14:40)



Background: Symmetric key cryptography

- Symmetric key cryptography: a shared secret key is used
  - Symmetric key algorithms are faster than public key algorithms
  - Mechanisms exist to agree on secret keys using public keys
- Message authentication codes (MAC) for data integrity
  - Alice sends message along with *tag = MAC(message, secret key)* to Bob
  - Bob can verify that message was not tampered with by regenerating tag, and verifying that it matches with received tag
  - If anyone tampered with message, recomputed tag will not match received tag
- Encryption for confidentiality with shared secret key
  - Alice sends *cipher = encrypt (message, secret key)* to Bob
  - Bob can recover original message with *decrypt (cipher, secret key)*

The second broad class of cryptographic algorithms are what are called symmetric key cryptography. That is the difference is that in public key cryptography, there were two keys. There was a public key and a private key. With symmetric key cryptography there is only a single key, a single shared secret key is used. And why this difference, why these two types of algorithms it is because the symmetric key algorithms are much faster than public key algorithms.

But of course, there is a trade off here, which is you need to somehow agree on the secret key, whereas public key cryptography can be used between any two people. As long as you know the other person's public key, which is publicly available, you can send some encrypted message. But with symmetric key cryptography, you need to agree on this shared secret key. And how is that done. There are various algorithms once again. You start with, Alice and Bob start with each other, public keys and then somehow agree on a secret key and then shift to symmetric key cryptography. So a lot of systems begin with public key cryptography then move to the faster symmetric key cryptography once a secret key is agreed upon.

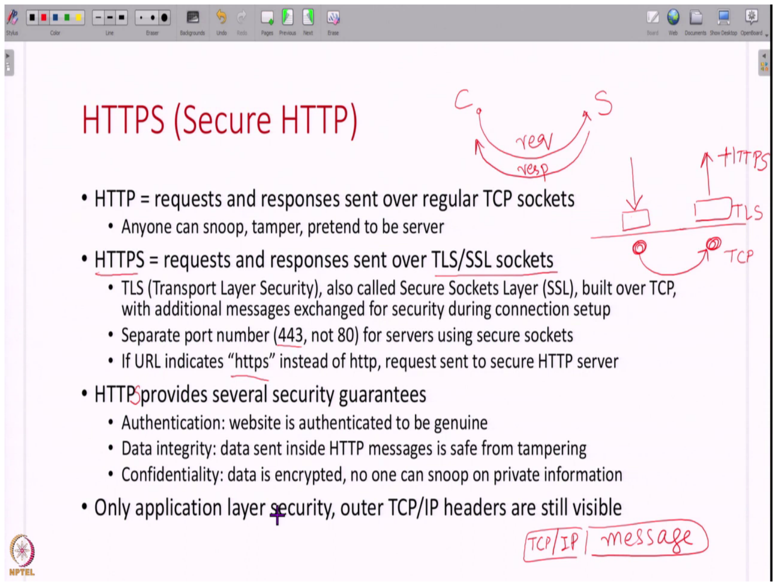
And some uses of the symmetric key cryptography are what are called message authentication codes for, once again for data integrity. That is, if suppose Alice is sending some message to Bob, along with the message Alice will send a tag, which is a message authentication code generated using the secret key that Alice and Bob are sharing. Along with the message, there is the tag. Now, Bob can verify that this message was not corrupted by regenerating the tag. Now, Bob gets the message. Now, Bob will once again using the secret key, compute the tag.

And then if this tag does not match the tag that came with, then something is wrong with the message I received. If somebody tampers with this message, then when you re compute this tag, it would not match, because two messages cannot, will not end up with the same tag. These cryptographic algorithms guarantee all of these properties.

So, if anybody tampered with your message, Bob will recompute the tag. It will not match the receive tag. Then you know that the integrity of the data was not preserved. But if this tag matches, then you know that nobody has tampered with your message. It is the exact same message that Alice had sent you.

And the other use for shared key cryptography is with encryption. Once again, you can do encryption with the public key cryptography also. And you can do it with shared key cryptography also. This is faster. But the principle is the same. Alice will instead of sending the message directly, Alice will encrypt the message using a secret key, send it to Bob, and Bob will decrypt the message using the same secret key. That is why it is symmetric key cryptography here. And all of these are in general much faster than public key cryptography algorithms.

(Refer Slide Time: 17:35)



HTTPS (Secure HTTP)

- HTTP = requests and responses sent over regular TCP sockets
  - Anyone can snoop, tamper, pretend to be server
- HTTPS = requests and responses sent over TLS/SSL sockets
  - TLS (Transport Layer Security), also called Secure Sockets Layer (SSL), built over TCP, with additional messages exchanged for security during connection setup
  - Separate port number (443, not 80) for servers using secure sockets
  - If URL indicates "https" instead of http, request sent to secure HTTP server
- HTTPS provides several security guarantees
  - Authentication: website is authenticated to be genuine
  - Data integrity: data sent inside HTTP messages is safe from tampering
  - Confidentiality: data is encrypted, no one can snoop on private information
- Only application layer security, outer TCP/IP headers are still visible

So, with all of these concepts in place now with the basics of public and private key cryptography, we are going to understand how secure HTTP or HTTPS works on the Internet today. So, any computer system that is dealing with any kind of private confidential, important data today runs on HTTPS. So, what is HTTPS?

So, we have seen what is HTTP. A client and a server, connect each other sockets and then the client will send an HTTP request, requesting some amount of some data, and the server will send an HTTP response, say the contents of a web page and the client will display that response on a browser. We have seen how HTTP works. And you use regular TCP sockets. Whatever data you write into a TCP socket is made into transport layer segments, and IP datagram sent over the network.

And these are just plain IP datagrams. If you have run wireshark, as I suggested in one of the previous lectures, you can just read through all the contents. You can look at all the bits in the packet and anybody on the network and just snoop all your packets and know all the content you are talking to that server. So, this might be okay if you are just consuming public content, like some publicly available news website, that is okay. There is no secret being exchanged here. That is okay.

But if you are doing secure transactions, like getting your bank information and all of that you may not want to everybody on the internet to know what your bank balance is, for example. So, therefore, in such cases, websites use HTTPS. What is HTTPS is there are some special sockets, secure sockets, which are called TLS stands for transport layer security, or SSL, secure sockets layer. These are just the same names for the same technology, two different names.

When you open these secure sockets, now the secure socket is just another software layer over the regular socket. They still run over TCP. But these secure sockets do not send whatever data is given to them directly. They will transform the data, run some extra logic over here before sending the data out over regular TCP sockets. So this is the TLS layer runs over the TCP layer and on top of this you will have HTTP. And this HTTP that runs on top of TLS is called HTTPS. It is the same request response protocol.

You send the HTTP request, get back responses, but you will send these responses via secure sockets, which themselves modify the message somehow, encrypt it and all of that and send the message over regular TCP sockets. And these servers that are using the secure sockets listen on a separate port number, not port 80, but port 443 for secure HTTP. And you might have noticed your URL also will no longer say HTTP, but it will say HTTPS, indicating that you actually are talking to a secure HTTP server.

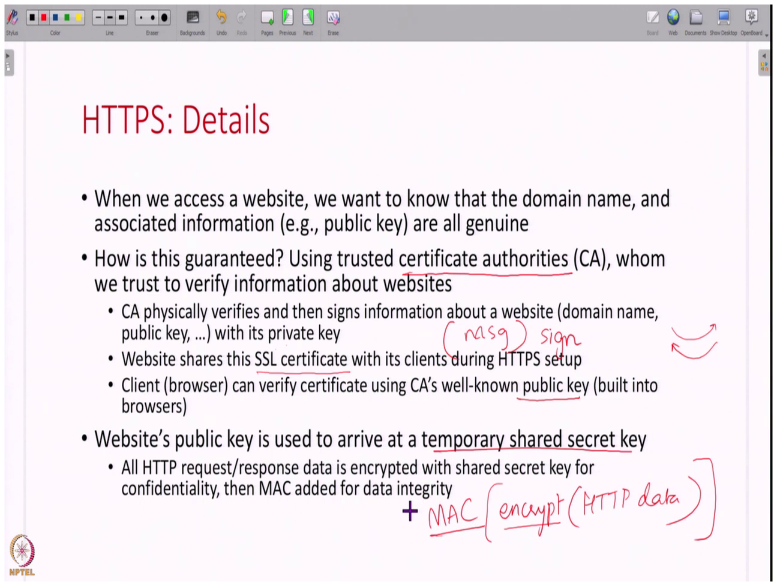
And what are the guarantees that HTTPS provides? The following security guarantees are provided by HTTPS, which is authentication. That is, if you are accessing a website, your online banking website, you know that it is actually the website of the bank. When you see this HTTPS logo, you usually see like a lock sign next to it indicating security.

So, it is a secure website. The website is authenticated. There is integrity of data. Whatever data you are sending over HTTP messages, nobody can tamper it. And the data is encrypted, so nobody can snoop on your information. So, confidentiality is also guaranteed. HTTPS provides all of these guarantees of authentication, data integrity, and data confidentiality.

And of course, all of this is provided only inside the message, application layer message that you are sending. Your HTTP request, response, that part is encrypted. Outside all the TCP/IP headers that you are adding, you will add the TCP header, then the IP header, Ethernet header, note that all of these headers are still visible. It is only the data being written into the sockets that is encrypted.

So, note that HTTPS does not solve all possible security problems. It only gives you what is called application layer security. The messages exchanged at your application layer are secure. But that is still of course a big step ahead. In addition to this, if you want security at the IP layer or at the link layer, you have to use other different solutions that we are not going to cover in this lecture. So, how does HTTPS guarantee all of these authentication, integrity and confidentiality and all of that? So, here are the mechanisms.

(Refer Slide Time: 22:29)



HTTPS: Details

- When we access a website, we want to know that the domain name, and associated information (e.g., public key) are all genuine
- How is this guaranteed? Using trusted certificate authorities (CA), whom we trust to verify information about websites
  - CA physically verifies and then signs information about a website (domain name, public key, …) with its private key
  - Website shares this SSL certificate with its clients during HTTPS setup
  - Client (browser) can verify certificate using CA's well-known public key (built into browsers)
- Website's public key is used to arrive at a temporary shared secret key
  - All HTTP request/response data is encrypted with shared secret key for confidentiality, then MAC added for data integrity

(msg) sign

+ MAC [encrypt (HTTP data)]

When you access a website, how will you know that the website, the domain name, and everything is all genuine? This is guaranteed using what are called certificates. So on the Internet today you have trusted certificate authorities. What these certificate authorities will do is they will verify information about the websites and give out a certificate. What is this certificate?

Suppose you are going to some website nptel dot ac dot in, this certificate authority will sign the information that, hey, this is the website, this is its domain name, this is its public key, all of this information about this website will be put in a message and this message will be signed using the private key of the certificate authority.

Remember, we have seen signatures before. Alice signs a message with its private key, then Bob can verify that the message came from Alice using Alice's public key. Similarly, the certificate authority will sign all of this information saying, hey, this website is genuine. This is the domain name. This is the public key. All the publicly available information about that website will be verified using the signature by the certificate authority.

So, these people we trust them to actually do some background checks. And they will give the certificates only to websites whom they have verified. They will not randomly give it out to anybody. But they, for a fee they will verify your information like your domain name. Actually, verify that your online banking website belongs to the bank and not to some attacker. They will do that verification, and they will sign that information with their private key.

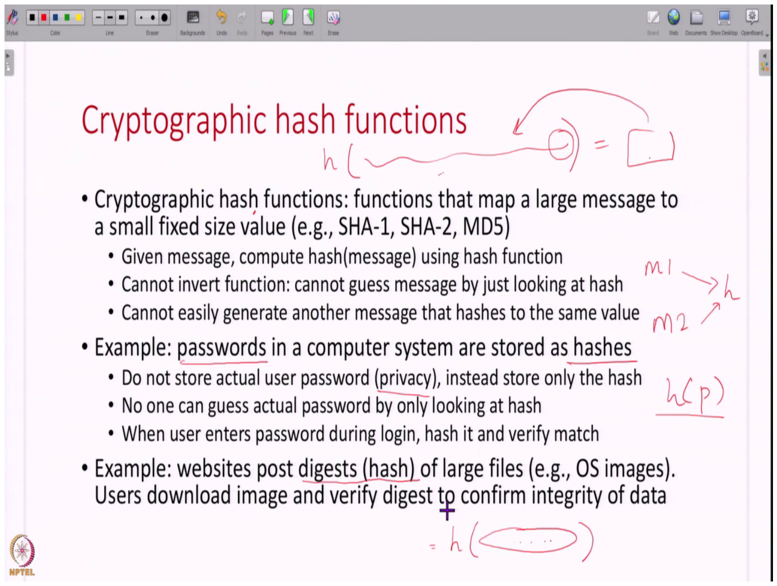
Now, when you as a user is accessing that website, that website will share that certificate signed by the private key of the certificate authority that certificate that it is called the SSL certificate the website will give it to you. During the HTTPS protocol has some additional handshake you know some setup phase where all of the security information is exchanged. In that phase the website will say, hey, I am genuine. Look at this SSL certificate I got from the certificate authority. Then the browser can actually verify the certificate using the certificate authorities public key.

Now, these certificate authorities are a few of them and their public keys are well known. We trust them. And that public keys are usually also built into browsers. So, using that public key, the browser can actually verify this SSL certificate that the website has produced. Therefore, now we are authenticated. We know that this website is secure, can be trusted. And once you know the website, the domain name is authenticated, you know the website's public key, then the client and the website will then run some algorithms to generate a shared secret key.

So, by just open communication, you can generate a shared secret key and nobody else who's eavesdropping knows that. There are algorithms to arrive at this secret key security. And once you arrive at a secret key, what will you do? Any HTTP request response that is being send, any HTTP data that is being sent, you will first encrypt it, so that nobody else can snoop on it. And then this encrypted data you will generate a message authentication code on this. And all of this you will send to the other side.

So the other side can decrypt your data using the shared secret key and can verify the integrity by regenerating this message authentication code. Together, you have ensured the integrity and the confidentiality of your information that is being exchanged. So, this is how HTTPS works. By first, the website will give you an SSL certificate, tell you that it is genuine, prove to you that it is genuine, then you will generate a secret key and you will use encryption and message authentication codes in order to guarantee confidentiality and integrity of the data.

(Refer Slide Time: 26:53)



Cryptographic hash functions

- Cryptographic hash functions: functions that map a large message to a small fixed size value (e.g., SHA-1, SHA-2, MD5)
  - Given message, compute hash(message) using hash function
  - Cannot invert function: cannot guess message by just looking at hash
  - Cannot easily generate another message that hashes to the same value
- Example: passwords in a computer system are stored as hashes
  - Do not store actual user password (privacy), instead store only the hash
  - No one can guess actual password by only looking at hash
  - When user enters password during login, hash it and verify match
- Example: websites post digests (hash) of large files (e.g., OS images). Users download image and verify digest to confirm integrity of data

So, the last concept that I want to discuss with respect to network security that is widely used in computer systems is this concept of what are called cryptographic hash functions. So, a hash function is nothing but a function that maps a large message into some small fixed size value. You take a large message, you hash it. There are many hash functions available like SHA-1, SHA-2, you might have heard of them or MD5. You take this large message and you hash it, you get a smaller message.

And what are the properties of this smaller message. You cannot easily invert it. That is you cannot guess what this message was by looking at this hash. So, for example, you cannot have things like, look at the last few bits of a message and use that as the hash that is not a good hash function. Why, because from that you know something about the message. But from actual hash functions, you cannot guess anything about the message by looking at its hash. They are, there are theoretical guarantees that give you all of these guarantees that you cannot invert, that is you cannot go from the hash to the message in any way.

Then the other property that these hash functions have is you cannot generate any other message that hashes to the same value. So, suppose a message m1 has hashed to some value h, you cannot easily find any other message m2 that also hashes to the same hash value h. Because of this hashes provide you a lot of security. For example, on all computers, passwords actually use hash functions.

On a computer for various users you do not store their passwords directly in a text file. You have to remember the password somehow to verify that the password is correct when the user types it. But you, what if you store the password and some attacker will read the password file that is very dangerous. You do not want to compromise the privacy of the users. So, what computer systems will store is they will store the hash of passwords. So, whenever the user sets a password, the hash of that password is stored.

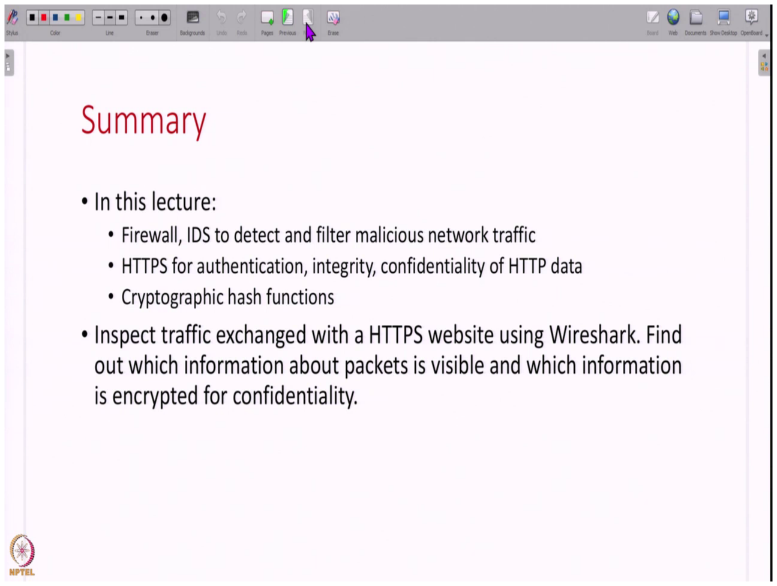
And the next time the user logs in, once again, you will take the typed in password, hash it and then verify with the stored hash. If it matches, well and good. That means the user has typed the correct password. So, this will ensure that there is some privacy of the user is preserved. You cannot guess what the password is by looking at its hash. At the same time, just because you are storing a hash does not mean any random person can log in. It is very hard to generate another password that also has the same hash. Therefore, it is still secure.

And another common usage is, websites, what they do is, they post this hash digests of large files. Suppose you are downloading some operating system image from a website, how would you know that the image has come correctly, has been downloaded correctly, nobody has tampered with it and all of that. What you will do is once you download this large message, you will take a hash of it and the website will tell you, oh, this is the hash you should expect.

So, if your data was corrupted, then the hash will not match. But if your data is correct, then the hash that you compute will actually match the hash that the website has already put up. In this way you can guarantee the integrity of large amounts of data that is downloaded over the internet as well.

So, with that, I would like to wrap up this lecture. I have told you a few brief concepts about network security, like what are firewalls, what are intrusion detection systems and what is HTTPS, what are cryptographic hash functions. So, this is only a very brief overview of network security. But these basics will help you understand some fundamental concepts when you design computer systems.

(Refer Slide Time: 30:34)



Summary

- In this lecture:
  - Firewall, IDS to detect and filter malicious network traffic
  - HTTPS for authentication, integrity, confidentiality of HTTP data
  - Cryptographic hash functions
- Inspect traffic exchanged with a HTTPS website using Wireshark. Find out which information about packets is visible and which information is encrypted for confidentiality.

So, to understand HTTPS better, I will request you to, once again, we have discussed wireshark based practical assignments in the previous few lectures also. Now, also, once again, go to HTTPS website and browse and see what all information about the website can you actually capture. You will see that the data itself you cannot see. Even though your browser is displaying the Wireshark cannot see what is inside the packet.

You can see TCP/IP headers, but you cannot see the application layer messages. Whereas if you go to a regular website, you can see all the data that is being exchanged. So, do this exercise to understand what security exactly does HTTPS provides to you. So, thank you. That is all I have for this lecture. And this week wraps up our discussion of how computer networks work. And in the next week we are going to move on to a new topic in the course. Thank you all.