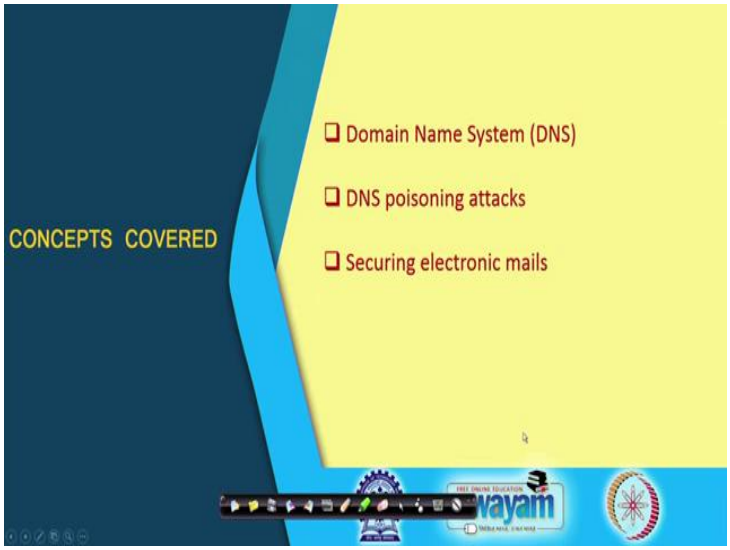
Ethical Hacking
Prof. Indranil Sengupta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 40
DNS and Email Security

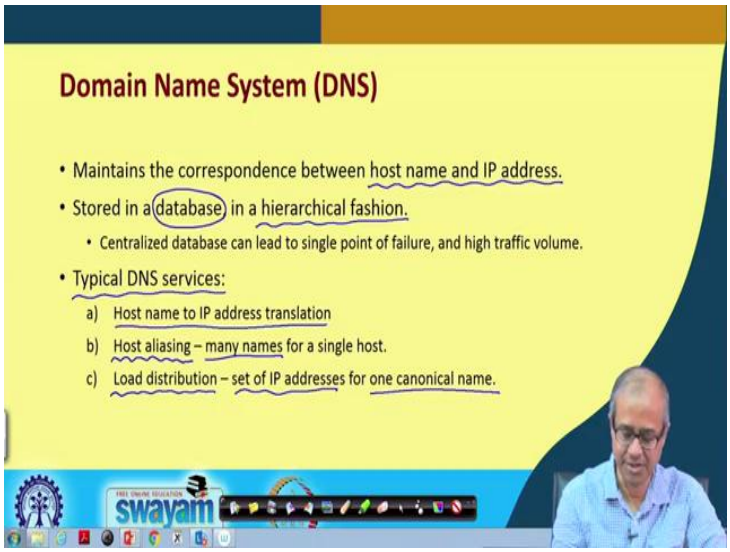
In this lecture we shall be talking about some of the security issues in name server and email servers. The title is DNS and Email Security. So, we shall be talking about security issues in domain name system (DNS) server and also in email systems.

(Refer Slide Time: 00:37)



These are the basic concepts that will be covered in this lecture DNS, some kind of attacks which uses DNS, like, DNS poisoning attacks and how we can secure electronic mail exchanges, ok.

(Refer Slide Time: 00:54)



Talking about domain name system or DNS, we all are familiar with what DNS is? On the Internet whenever you want to access some resource, we have a name or a URL. Using the URL like for example, $www.google.com$ that is the example of a URL, but before we can send some requests to $google.com$ we have to find out the IP address of the corresponding machine. Only then a packet can be sent. So, DNS server helps us in retrieving the IP address from a given name or domain name, ok.

So, it maintains essentially the correspondence between host name and IP address. Each host is given a name. I have given example $google.com$. For example, at IIIT Kharagpur, our server name is $iitkgp.ac.in$. That is the name of our server, web server, ok. And this information host IP address, host name and IP address correspondence, this is stored in a database. What kind of a database? You see, DNS is a server program, domain name system that DNS maintains a local database with it and in that database among other things this host name and IP address correspondence, this information is stored, ok.

And this database is not a single database, because all information cannot be made available in one place. For example, in your DNS server which you have installed, you cannot have the information about all the domain names in the entire world. There are millions of such names, right. So, you cannot have everything here. So, there should be some kind of a hierarchy. You first try to answer, if you do not have the information ask

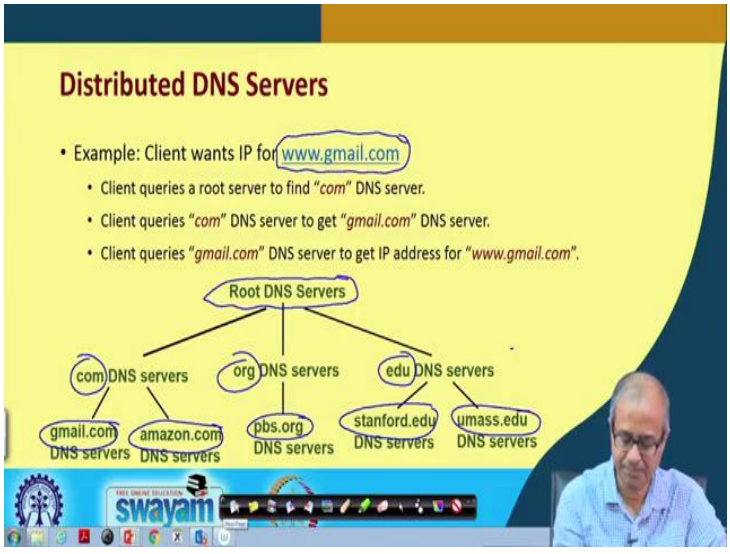
someone else. That someone else says if does not know, he will ask someone else again. Like that there will be some kind of a hierarchy, ok.

So, in a hierarchical fashion the database is maintained, because as I said, if you had a single centralized database where everything is stored in one place, firstly, it is very difficult to maintain and secondly, if that central database fails then everything fails. You cannot use anything, ok.

Now, DNS server provides typical services like these. Like, the most important I told you, is the host name to IP address translation. Given the host name, it will give you the IP address and it supports something called host aliasing. Like, a single host may be having various names. You can have multiple names for a single computer. If you want, if you pay for the domain names, you can have multiple names registered. Also it supports load distribution.

Load distribution means, I give an example of $google.com$, but $google.com$ does not refer to a single machine, there are large number of machines which are maintained by Google. They are all having the same name $google.com$. There are a set of IP addresses for a single canonical name like $google.com$. These servers too have some kind of load balancing. Suppose, I am sending a request to $google.com$, I will be contacting my nearest server. If there was a single server, everyone was accessing that. That server would have been really overloaded, right, ok.

(Refer Slide Time: 04:59)

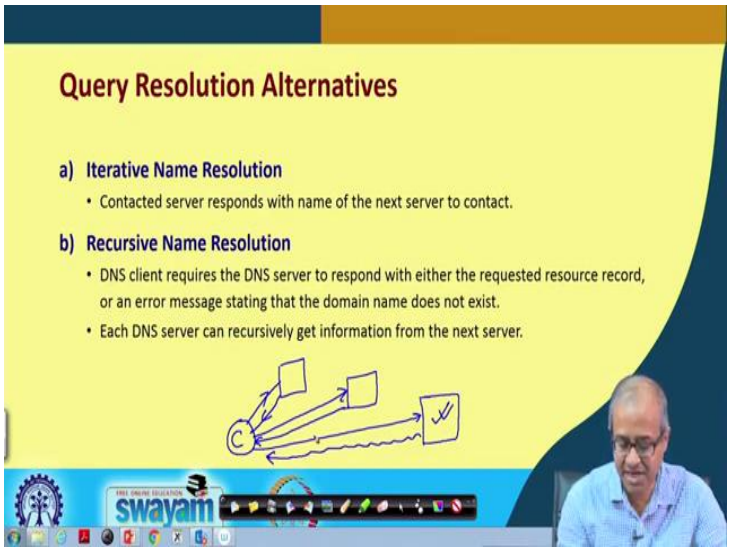


Now, distributed DNS server, as I told you DNS is not in one place, it is some kind of a hierarchical structure. Let us give an example. Suppose, we want to access $gmail.com$. When I say $gmail.com$, from the right side com then gmail in that order these are the names of some DNS server; com refers to a DNS server in the $.com$ domain, gmail refers to one DNS server under that com that refers to the $gmail.com$.

It starts with a route. Initially, there is a, we refer to as a root DNS server. So, everything is under root. Under root the first thing here this com, there can be com, there can be org, there can be edu, there can be so many other things. Like in India, we know $.in$, country domain names, large number of such. I have just only showed 3. Under com all websites which end with $.com$ they will have their own DNS servers. They will all be located.

For example, there will be some DNS server for $gmail.com$, DNS server for $amazon.com$ and so on. For org, I have given one example $pbs.org$; edu, $stanford.edu$; University of Massachusetts, $umass.edu$ and so on. So, this kind of domain name server, DNS server is not one. There can be multiple and I have just showed 3 levels. There can be large number of levels in this hierarchy, ok. Under the Stanford there can be several departments, each of them may be having their DNS servers and so on, ok.

(Refer Slide Time: 07:01)



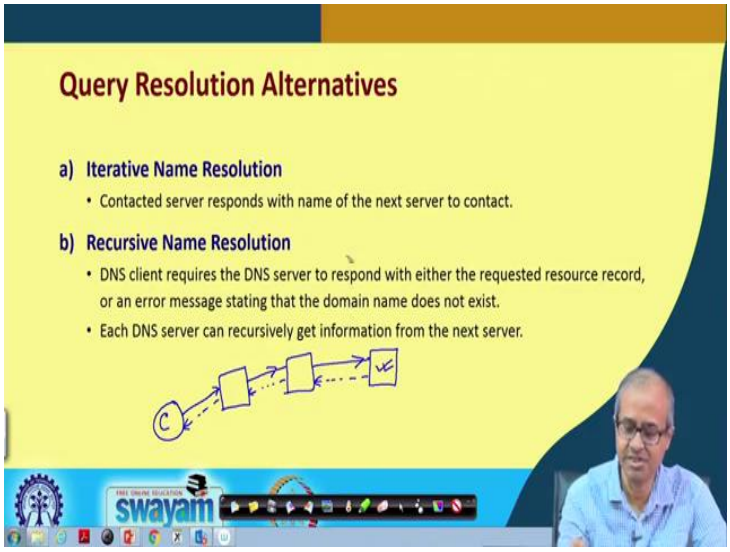
Now, when a client sends a request to a DNS server, let us say, it requires that I have this name $gmail.com$ give me the IP address. Now, there are two alternatives. First is called

iterative. Iterative means suppose this is my client machine, ok. This is my client. Client is sending a request to the nearest name server. Nearest name server replies back to the client saying that well, I do not have the information, but please connect the next level root server, next level DNS server.

So, it sends the request to next level. Again a request may come back saying that well I do not have the information, please send the request to this DNS server and finally, this DNS, this DNS server might have the information in the database. It sends back the correct response. This is called iterative.

The client sends multiple requests to multiple DNS servers one by one, depending on the response it has received from the previous one. So, the client has to handle a large number of requests and responses, right. This is called iterative name resolution. But recursive name resolution is slightly different. Here the load on the client is less.

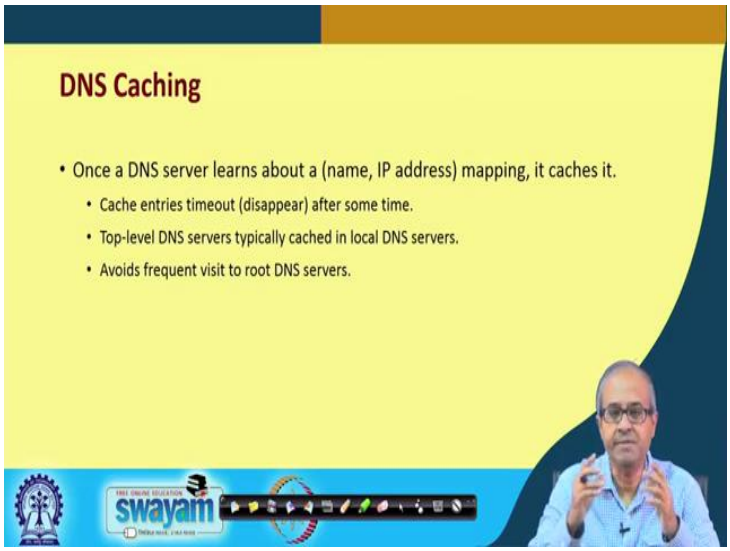
(Refer Slide Time: 08:26)



So, what happened is something like this. Suppose again this is the client. The client is sending a request to the nearest name server. Name server may not be having the information, but instead of sending the request directly back to the client, it sends the request to the next high level name server and next high level server sends the request to the next high level server.

So, the client is not disturbed. And once the result is found then the response follows the same path and finally, it will come back to the client. So, to the client as if the client had sent one request and received one response. But internally the name servers were working among themselves recursively to resolve the domain name. So, this is what is mentioned here, ok, fine.

(Refer Slide Time: 09:39)

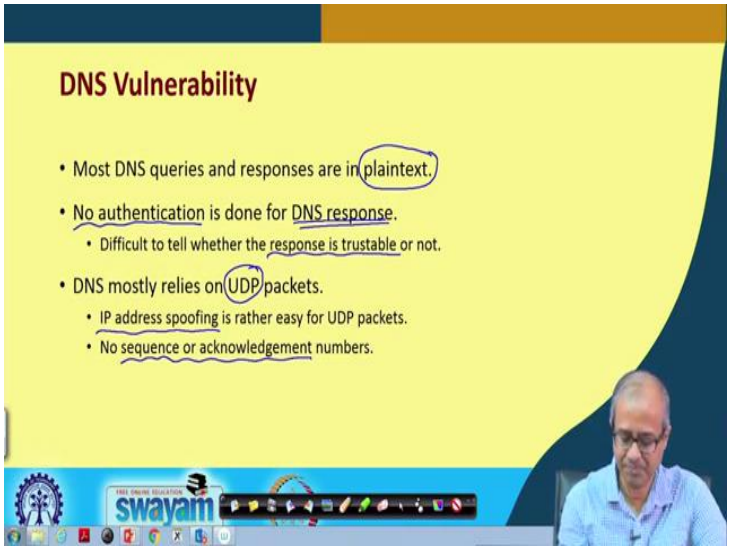


DNS caching is one important. You see when you request for a, for something to a DNS server, DNS server may not be having it in its table, so it asks some other DNS server and gets back the result for you. But, what it does in addition? It enters that information in this local table or a local temporary memory called cache, DNS cache, because it expects that you may send that request again in the near future and if you send, it can send back the response directly from the cache. It will not again send it to the next high level root server. That is the purpose of the cache. So, DNS caching is something which is very common. It is used to increase the efficiency of operation, ok.

So, it is mentioned here once a DNS server learns about a name IP address pair, it puts it into a cache, a fast memory, a portion of the memory. It puts it there. And these are not permanent entries. They have some timeout, maybe a few hours, few days, 1 week. After that they will automatically be removed from the cache from the memory, right. So, the requests that got resolved in the higher level DNS server usually those information at cached in lower level servers, so that when future requests come, they can be resolved at

that level itself, ok. So, this will avoid frequent visit to the higher level DNS servers, this is the advantage.

(Refer Slide Time: 11:24)



But with this advantage comes some other problems also, which we shall see. Talking about DNS vulnerability, the first thing is that all DNS requests and responses which go and come back they are in plaintext. Like, if you install a packet sniffer somewhere in the network and if you observe all the packets, you will also see the DNS request packet and DNS response packet; you can find out everything what is going on, ok.

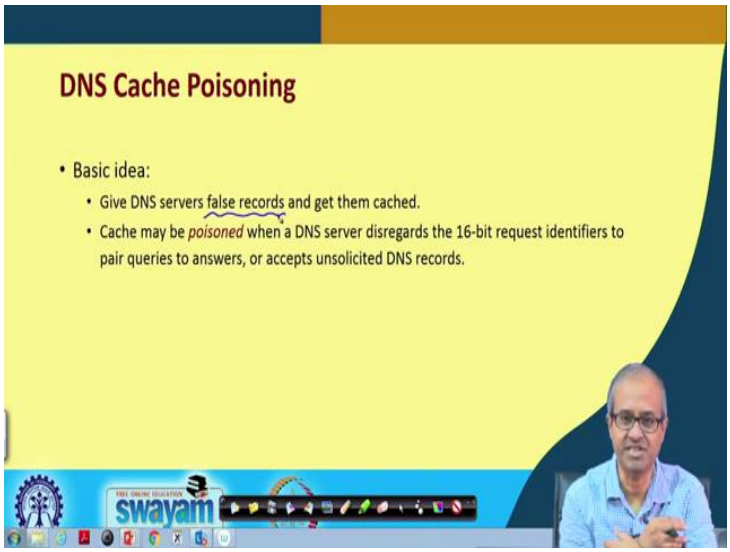
And secondly, the normal version of the DNS server that is used, no authentication is carried out for the DNS responses. Like, suppose you are a DNS server, I send you a response, I send you a request, based on my request you send me back a response. I do not check whether the response is actually coming from you or not, which means I am not authenticating you. This is what happens for the normal DNS servers that are used. When a response comes back, I accepted in good faith that well whatever they asked for, I have got back the result, ok.

So, it is difficult to tell whether the response is trustable or because it is always possible that someone is sending me a fake response not the correct response. Let us say, I asked for state bank of India IP address, $sbi.in$ let us say, but instead of the original SBI IP address, someone is sending me some other IP address so that instead of going to the SBI site I am possibly going to a hacker site and I am in trouble. Maybe I will be getting a

page which looks very much similar to this, to the state bank of India website. I can type in my username and password and well everything is gone. The hacker can steal all my money using that information, right, ok.

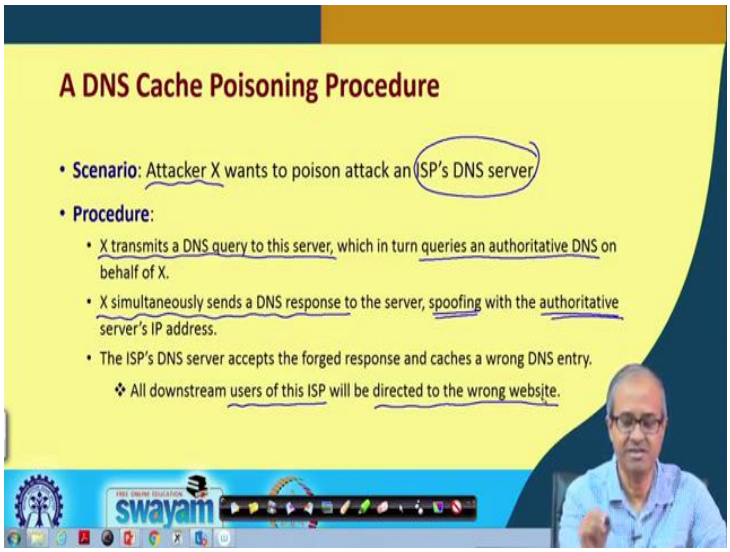
And the other thing is that DNS requests and responses. Because they are very small packets, they mostly rely on the UDP protocol. They do not relay on TCP, because UDP is much faster. And on UDP packets, IP address spoofing is much more easier, because unlike TCP there is no sequence number or acknowledgement number. In TCP, it is difficult because all successive packets of a message are assigned successive sequence numbers. So, just by looking at the sequence numbers you can know whether things are proceeding in, proceeding in the correct order or not, ok. But in UDP you cannot do that.

(Refer Slide Time: 14:30)



Now, DNS cache poison is something which is you can say, a kind of attack and this arises because you are using DNS caches. You are caching the requests. Basic idea is that as I have said whenever DNS server received some response, it puts it in a cache. But if you deliberately send some false records, wrong IP address that will also get cached. This is what we are saying that we are poisoning the DNS database. So, in the future, if some requests come, wrong IP address will be returned to the requesters, ok. This can be a security problem.

(Refer Slide Time: 15:19)



Now, one mechanism for DNS cache poisoning I am just stating here. Let us say, the scenario is, there is an attacker X, ok. Sorry, there is an attacker X, who wants to poison some DNS server, let us say, ISP's DNS server. My nearest ISP's DNS server I want to poison it.

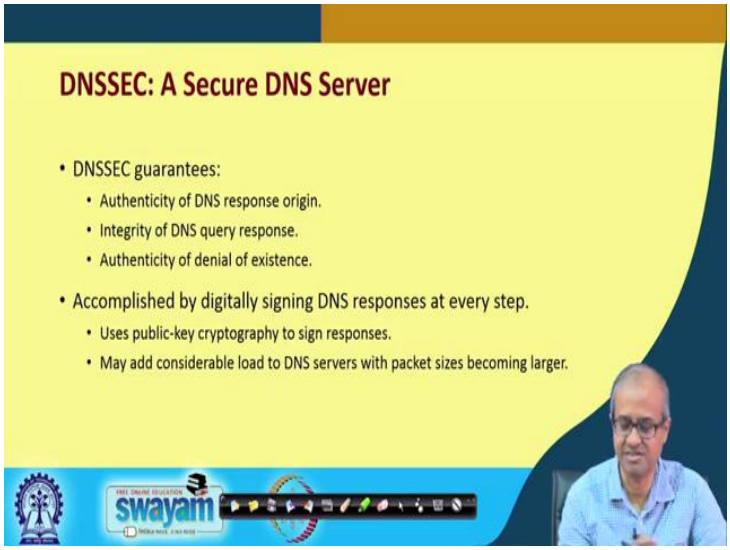
So, what I can do, X will transmit a DNS query to the server. It will send a request that well, I want the IP address of $sbi.in$, state bank of India. So, what the ISP's DNS server will do? If it does not have the information, it will ask the next higher level server for the information, it will in turn query some authoritative DNS server, high level to get back the response. But what the attacker will do is, immediately after sending the query the attacker also sends back a corresponding DNS response.

So, the ISP's DNS server may feel that well I had send a request to the root server that response is coming back now. So, for that purpose what X does, it spoofs the IP address with the next higher level DNS servers IP address. So, when the response, the fake response goes to the to the ISP's DNS server, the DNS server believes that it is the correct response and it puts it in the cache. But it is a wrong IP address, ok.

Subsequently, all users who are using that particular website or that particular domain name will be redirected to this wrong IP address, wrong website. This is what is meant by DNS cache poisoning and this can be done using this mechanism. There is a secure

version of DNS, called DNSSEC. Well of course, in this secure version all these things are not easy to do. They are much more difficult.

(Refer Slide Time: 17:39)

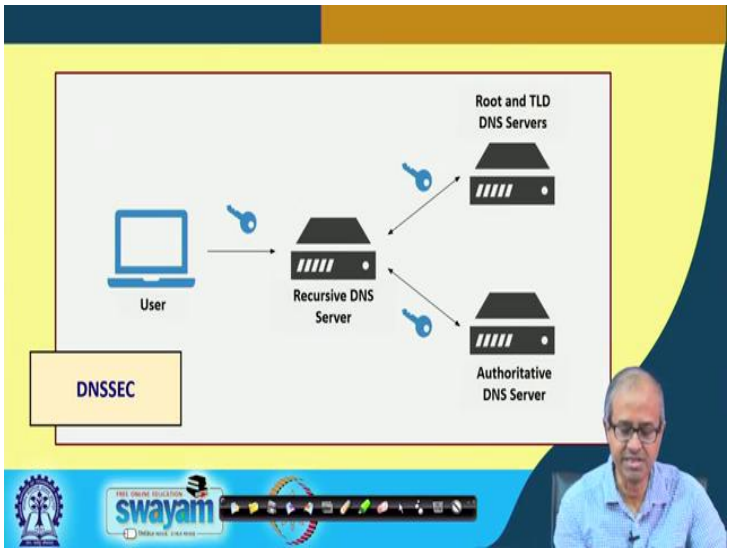


DNSSEC provides some additional features or additional capabilities, like it authenticates the origin of the DNS response, who is sending the request, it authenticates. It also checks the integrity of the query response. Whether the DNS query was modified in transit? That is called integrity, data integrity and authenticity of denial of existence. If someone says that I have not found it, that also there has to be some kind of authentication behind that. There is some mechanisms; I am not going into detail.

So, here at every step there is some kind of a digital signature process which is going on. Using this at every step you are verifying whatever you are getting is authenticated by the proper sender. In that way you can be sure that you are not receiving any fake messages or fake responses. But the downside or drawback is that lot of additional calculations or computations are required to do this, lots of encryption, hash function calculation etc.

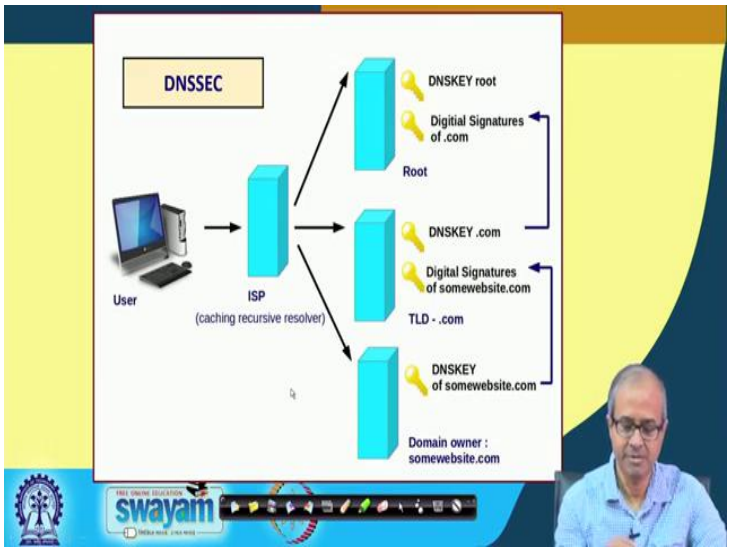
This may add considerable load on the DNS server and also the packet sizes may become larger, because now the packets will carry many more, many extra information with respect to this authentication, other things, ok. These are the drawback. So, pictorially I am showing this.

(Refer Slide Time: 19:23)



This is the user, this is a DNS server and this key symbol shows that all these links are authenticated links. So, when user sends a request, this DNS server verifies the authenticity of the users. When this DNS server sends a request to the higher level server that is also authenticated both ways, here also both ways. So, at every step, we are guaranteeing that fake messages are not getting transmitted or recorded.

(Refer Slide Time: 19:59)



So, another picture here; suppose, this is my ISP, user is sending a request. An ISP, if it does not have the information, then it requests the higher level. At every level, there is a

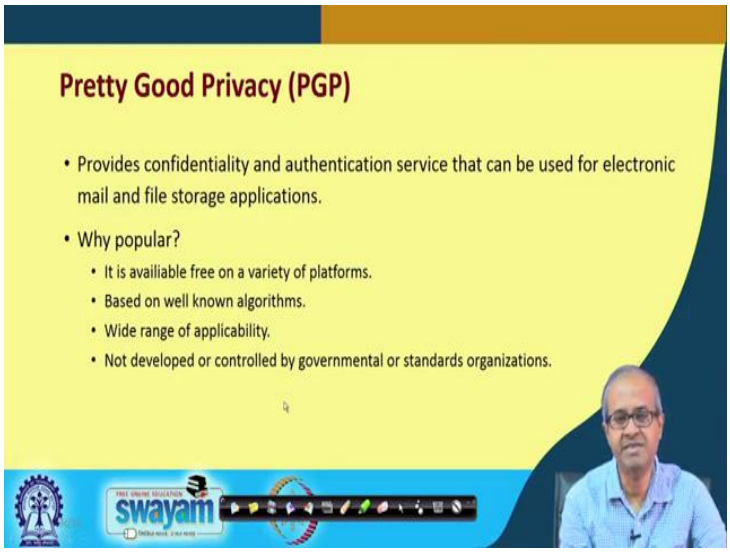
digital signature process going on. So, I am showing it using this DNS key of some website dot com. Using that key it is digitally signing the response and sending back here. This is digitally signing and sending back here and finally, it will be coming back.

So, whenever using recursive resolution one name server is contacting some other name server, all messages are digitally signed by some secret key, by some kind of a, some kind of private key. So, this private key, public key pair, this kind of things has to be there in this mechanism. I am not going into the detail exact mechanism, but roughly speaking, this is what is happening, ok.

(Refer Slide Time: 21:01)

Now, coming to email security, just one example case study I am looking at.

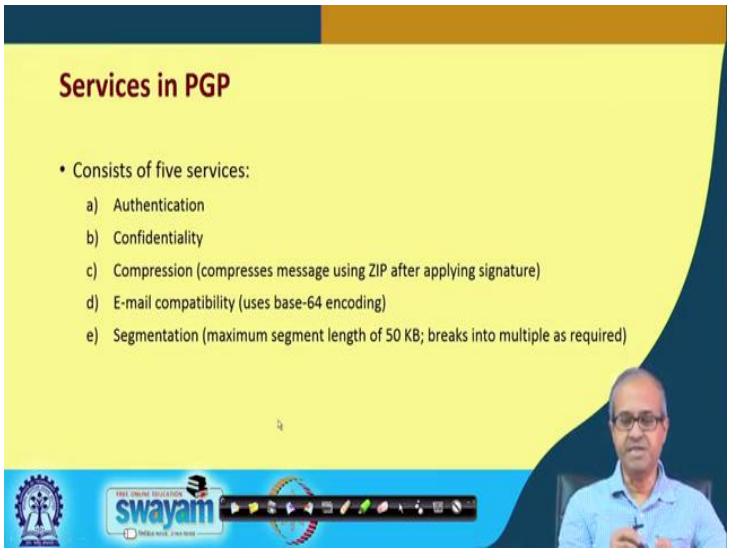
(Refer Slide Time: 21:07)



There is a secure mail system called PGP, pretty good privacy. And PGP, you see normal email service does not provide you with any security, there is no confidentiality, no authentication, nothing is provided. But if you have PGP installed on top of your mail server, you can have a host, lot of services in addition like confidentiality, your mail will get encrypted, no one can read it, authentication - you can verify from whom the mail is coming, and these will serve towards having a secure email transport mechanism.

PGP is quite widely used, because it is available on a, on a wide variety of platforms. The algorithms that are used are pretty well known so that you can be confident about their operations. It can be used in variety of scenarios and this is not controlled by any central organization. You can use it for your own purpose, if you want.

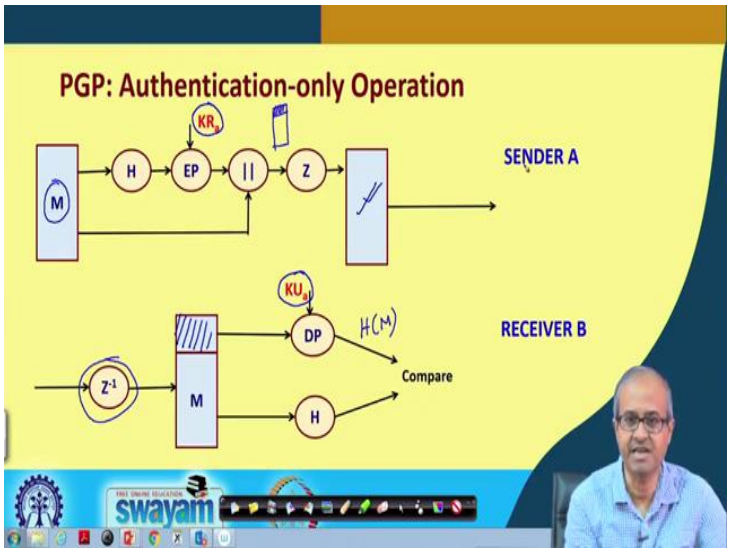
(Refer Slide Time: 22:21)



The kind of services PGP provides, are broadly, there are 5 services as you can see. First is authentication. It will verify the origin of a message from whom a message is coming. Confidentiality, it can encrypt the message. Compression, sometimes it compresses some mail using some well known compression algorithm, zip it, uses zip, zip. Zip is a well-known compression, you may be knowing and for email compatibility it uses some encoding this is called base 64 encoding.

But those of you who know about the SMTP protocol for electronic mails; you may be knowing that all electronic mail attachments whatever you are sending they are encoded in some form. This is called base 64 encoding. So, it is compatible to that. And segmentation means, if your message size is more than 50 kilobytes, it will break it up, each chunk can be maximum 50 kilobyte that is how it works.

(Refer Slide Time: 23:25)



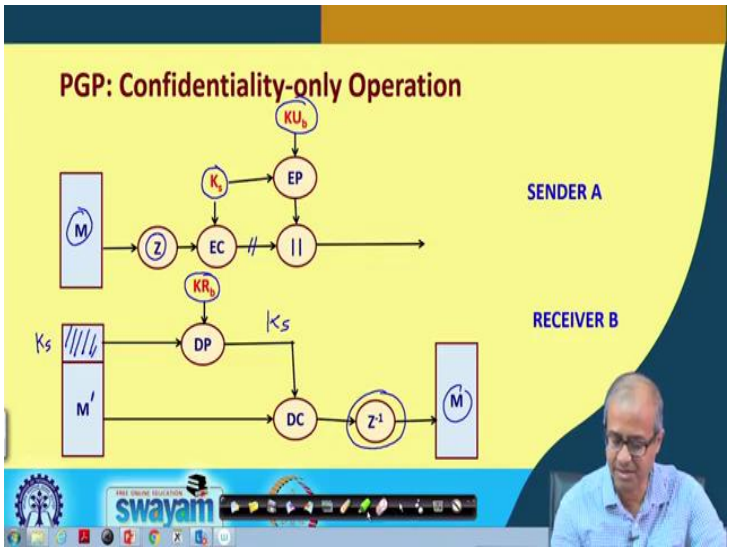
So, let me show you two typical scenarios. In PGP, suppose I want authenticate. Only authentication, I want to verify that the message is coming from a sender. Something is happening at the sender side; some something is happening at the receiver side. So, the top diagram is from sender side; bottom diagram is for the receiver side. Let us try to understand. The sender is trying to send a message. This M is the message. So, what it does? It will hash. H is a hash function, cryptographic hash. You have studied sha-1, sha-1, MD-5. So, many methods are there. So, it applies a hash on the message. And EP is, it is an encryption process. It encrypts the hash value using KR, means private key of sender A.

You recall, I mentioned earlier when using public key cryptography you need authentication, you will have to encrypt something using the private key of the sender. That is why the sender is A. I am encrypting it using the private key of the sender and then I am concatenating it with the message; that means, whatever the message was, the message remains and I, this encrypted hash code also I add with it. Then finally, I compress, Z means compress, zip and that compressed version is sent over mail to the receiver. So, receiver will receive this.

So, what the receiver will do? First thing is that, because it was compressed, it will uncompress it first. It was zip, there is a command unzip reverse, it will unzip it. After unzipping it, it will get two parts, one is the message, one is the encrypted hash value

part. What it does? It decrypts the encrypted hash value using the public key of the sender. Public key is known to the receiver. It gets back the hash of the message. So, here you get $H(M)$ which was calculated earlier. You repeat that calculation based on the message $H(M)$ and you check whether these two are equal or not. If they match, then you conclude that your message is actually coming from A. This is how authentication is carried out, ok.

(Refer Slide Time: 26:14)



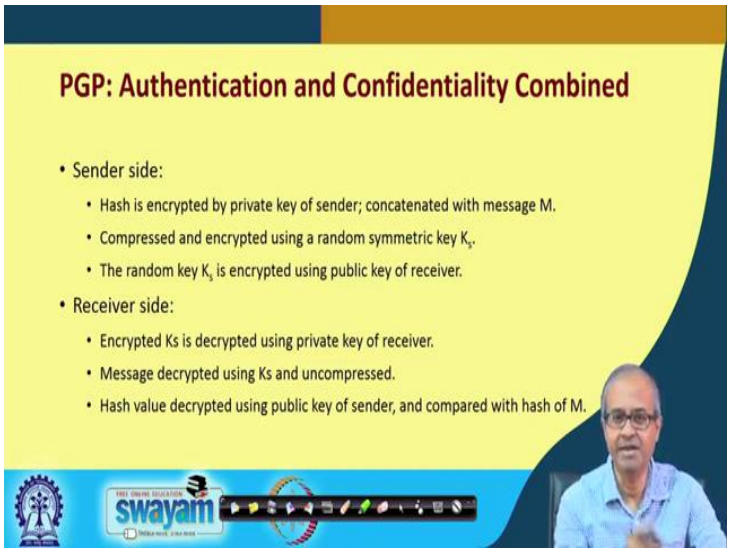
And the other thing is confidentiality. Suppose I want only confidentiality. I want no one can read my message. How I can do this? So, here again there is a sender A and the receiver B. Sender what it does? It wants to send a message. Now, in PGP, first a compression is done, zip, that is how PGP works. First you do a compression, then you generate a random symmetric key, K_s is a symmetric key that we generate randomly. And you encrypt this compressed version of the message using a symmetric key algorithm like AES.

So, this is your encrypted version of your mail body. But this K_s must also be known to the receiver. So, this K_s you are now encrypting using public key cryptography, using the public key of the receiver B and you are sending. So, what you are sending, receiver is receiving one, is actually whatever you are receiving is, this is not exactly M, this is the, you can say encrypted version of M. You call, you can call it M', encrypted version of M and here you have the encrypted version of K_s. These two things are coming to you. So,

here you do a decryption first using the private key of B. You get back K_s. Now using K_s you can decrypt this M' and it was compressed in the beginning. So you uncompressed at the end. You get back M. This is how it works, right.

Now, these two examples I showed, this refer to how you can do authentication, how we can ensure confidentiality. Now, both these things can be combined together also. I am not showing you the detail diagram. Just I am mentioning that you can have a combined service also, authentication and confidentially combined.

(Refer Slide Time: 28:25)



Whatever I showed, you can put them one by one in sequence. Like on the sender side, what you can do? You can compute the hash of the message, hash value you can encrypt by the private key of the sender for authentication, concatenate with message M. Just like what you did for authentication. You compress it, encrypt using random symmetric key. Now, whatever you did for confidentiality and the random key itself is encrypted using public key of receiver whatever you are doing there.

And receiver side encrypted K_s is first decrypted using the private key of receiver, then with that K_s you decrypt the message and compress it, you decrypt the hash value and compare with the compare with the hash value of M. If it matches, then you say that it is authenticated; otherwise it is not authenticated. So, in PGP you can have all this kind of service which will make your email service much more secure and also you can verify from whom your main messages are actually coming, ok.

So, with this we come to the end of this lecture where we have talked about security issues in two of the very important protocols that we use in our daily life. One is the name server, DNS servers and other electronic mails, ok. In our subsequent lectures, we shall be discussing some other aspects of security and protection, how we can safeguard our systems. This we shall be discussing in our next lectures.

Thank you.