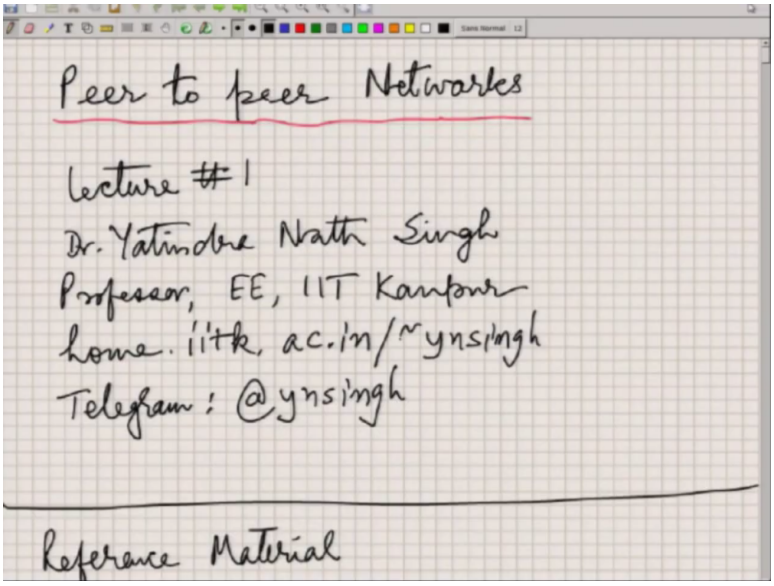
Peer to Peer Networks
Professor Y. N. Singh
Department of Electrical Engineering
Indian Institute of Technology, Kanpur
Lecture 1
Introduction to Peer to Peer Networks

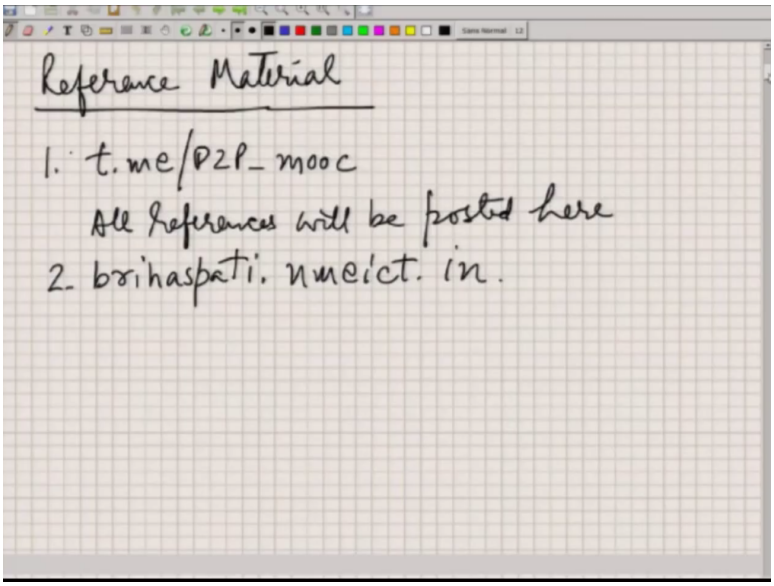
Welcome to the course on Peer to Peer network.

(Refer Slide Time: 00:19)



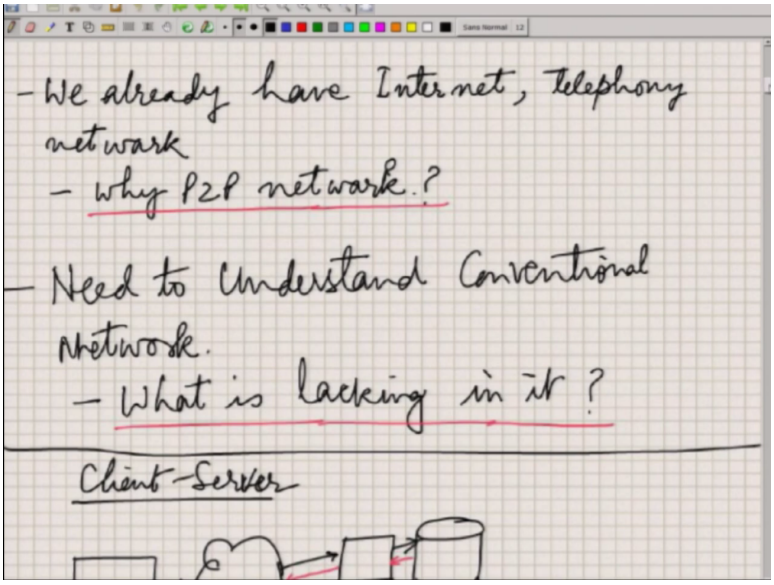
I am offering this course for the first time as a MOOC. This is lecture number one, where I will be introducing you to the various basic things about Peer to Peer Network, and as we go along, we will explore more and more about this. For my introduction, I am Professor Y. N. Singh, and I am in IIT Kanpur. This is a website where you can find information about myself, and this is my telegram ID over which you can communicate to me if you want.

(Refer Slide Time: 01:00)



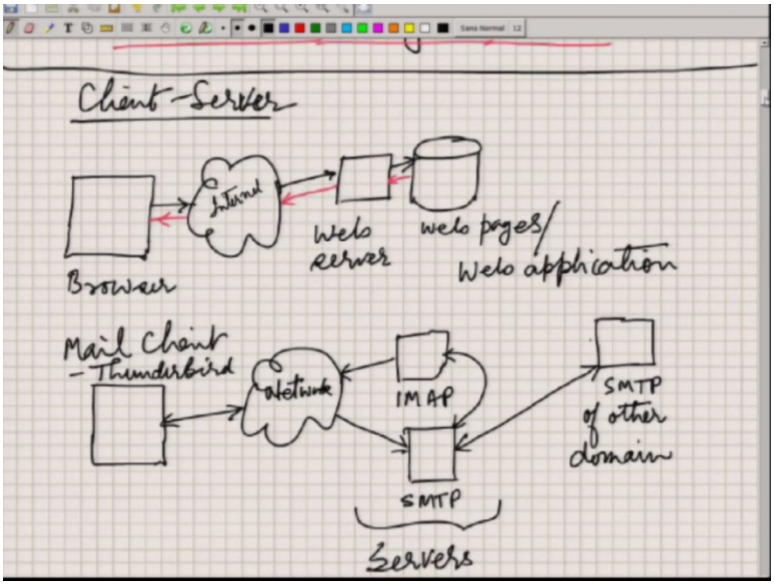
This course's reference material will be available at the telegram channel with this string **t.me/p2p_MOOC**. I will be posting all my lecture notes. All video recordings which will be made here will also be available there. Some of the material is available at Brihaspati.nmeict.in, where I have pushed in the lecture notes or the study material used in my courses. The same course variants have been taught in IIT Kanpur for in regular semester for PG students.

(Refer Slide Time: 01:45)



Now let us come to why we require this P2P network. We already have the Internet. We already have a telephony network; we are able to communicate with each other, and we still talk about Peer to Peer networks. And what is this Peer to Peer network is actually? So for this, we need to understand what a Conventional Network is? The Internet we so-called or the Telephony network and what is lacking in them, and because of that gap we are, we apply trying to go to Peer to Peer systems.

(Refer Slide Time: 02:24)



If you look at the conventionally or what happens typically nowadays, we call it client-server technology. There is always a server which you can all identify by their names. So, for example, you go for checking your mail on Google. So you will still identify www.gmail.com as the name of that server. This, somehow, on the Internet, will get converted to another address of the server machine. You will be connecting to this server from your machine, laptop, desktop, browser, or mobile phone.

Your browser's program will mostly communicate with this www.gmail.com will mostly be a browser. And this browser will be sending requests over the Internet, which you can see here. So this is the Internet. This will go to the webserver, which in turn will talk to an application, which you can see here, this application can be flat web pages or can be some logic, which is dynamically generating web pages for you, and with the red path, you will be returning to the

browser. And that is how when you want to view an email, so a command goes all the way to the web server, and then a new page will come back, and it will show what your email id is.

Here we are slightly more sophisticated because we run a small application in the browser, which talks to a web server, fetches the information, and continuously displays it to you. Similarly, we can think of another application called mail. So we all have been sending emails. So the client which I use on my desktop or in on my laptop is Thunderbird. You might be using a Gmail client on your mobile, android mobile phone, or you might be using something else. So, in that case, these clients will generally talk to the IMAP server (Internet Message Access Protocol), Mail Access Protocol, an SMTP (Simple Mail Transfer Protocol) servers.

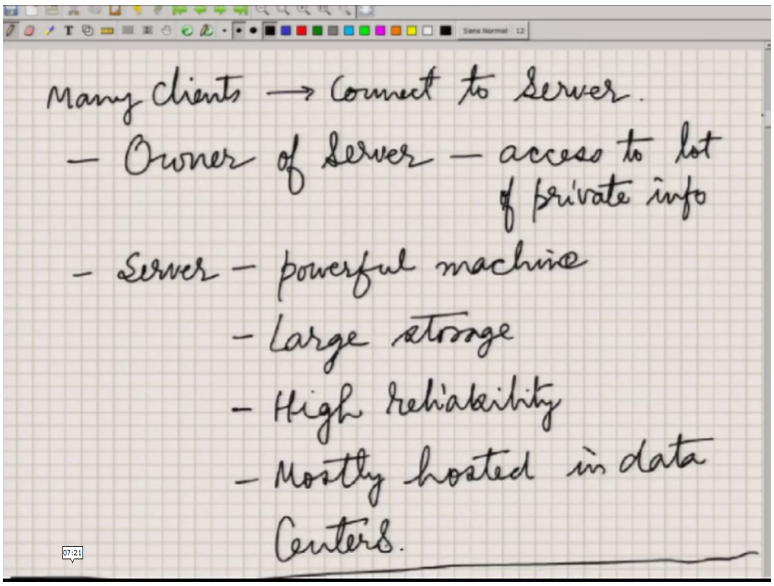
So when you want to send an email, your client will use this SMTP protocol. The protocol is a set of rules, which have to be followed by the two endpoints or two sides across the network to understand each other's language. So there is a set of rules which have to be framed for communication. This is what we call protocol, and these are formally documented. For receiving the email to the client, you use IMAP connect to an IMAP server; for sending it, you again give it to an SMTP server.

So, but the server is very sacrosanct; they have to be available 24 *7.They cannot go down, and they have to have an enormous capacity so that they can serve a large number of users who are connecting from their laptops or desktops or mobile phones. And of course, in the case of SMTP servers, we even defined zones so that IIT Kanpur will have one SMTP server and one IMAP server.

And we talked to IMAP, the SMTP server of other institutes for transacting the mails. And whenever this client talks to the SMTP server, they will do authentication so that when you are sending a mail, we have a record; you logged in as which particular user, when you have given

send the mail through the SMTP server. This is used if we need to investigate whom the original person has submitted the mail. So this is kind of a security system.

(Refer Slide Time: 06:29)



This means we will have machines on the net, and so these, there will be many clients that will connect to the server. That is one of the key things here. So there can be thousands of clients who can connect to the server. Now here you should think if Google is also running a mail server www.gmail.com on a browser, there will be billions of people trying to connect to Google at any point in time. A single server probably cannot handle this.

They usually use a not one but multiple servers, and different people will always connect to different servers. So maybe everybody who is in the UP area is connected to one specific server, somebody else is connected to some other specific server, they essentially do partitioning to create, to support a large number of users. And now one of the problems with this server-based thing is that owner of the server will have a lot of private information of yours.

Google, as of now might be holding, knowing about your phone number, they know about, they will not be knowing about your password because when you send a password, they will immediately convert into a hash; I will type, I will talk about what the hash is, as we go along the course. So hash is Layman's term for now. As you take any string, you pass it through the hashing function; it will give you a certain number of bits. And if you know these bits, you

cannot get back the original information from which this was generated. But if you know the original information, you can find out the hash. So it is a one-way transfer, one-way mapping to from an infinite set to a finite set.

Usually, all passwords are being hashed, and this hash value is kept in a table. The server owner will not be able to find out what your password is, but if you give the password, he can compute the hash and match the hash and make a guess that yes, the password is correct. Yes, there is a finite possibility that more than one passwords may map to the same hash value, so if this collision is what we call collision, the chances this collision happens is extremely low. So most of these devices work on that principle. That is one problem with the server-based system. So a lot of this private information is available. And if somehow a compromise happens on the server or the owner itself goes rogue actually, then they will have much information about you.

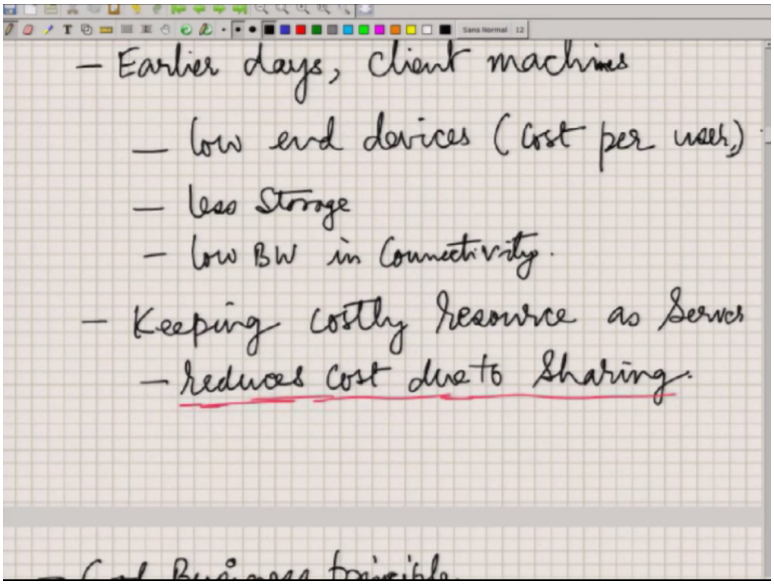
Normally, users would like to have the right over the information about them. They do not want anybody else knows. So it should be the right of the user to tell who should know how much about him. So that right to privacy is important. So, this will typically be handled by the government's regulation, which will apply to the server owner. Of course, we do get into trouble if the servers are residing outside India when the Indian government cannot control the owner.

We require that servers of any Internet service, which has been provided in India, any kind of application, the servers have to be maintained within India so that the information is not outside. And the government of India can control or access that information on a need basis. So the law allows this. The second problem is that these servers need to be extremely powerful machines, and they need to have huge storage and need to have extremely high reliability.

Usually, this is done by running multiple servers, and they act like hot standby. Even if one server fails, the other machines automatically take over. We have those kinds of mechanisms to be built. And because you want high reliability, good air conditioning systems, so significantly, all this will essentially be set up in data centres. So the cost of running a data centre will also be huge. This essentially entails many infrastructure costs if you are going to go with the client-server thing. But this is required in the past because good computing was costly. People had low cost, low computing power, low storage, and kind of devices. So anything which is costly was shared among a large number of people. That is where the server concept came in.

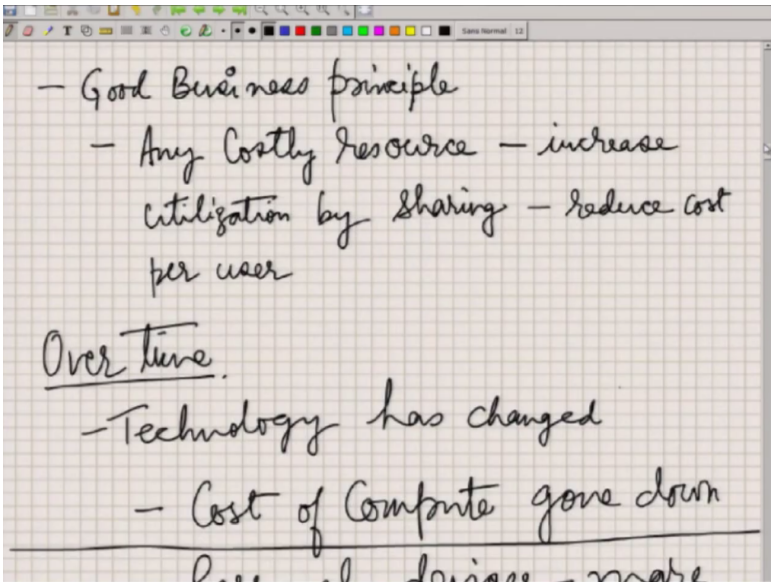
Initially, there used to be a compute server, so used to do Telnet. Telnet is a program by which you can log in to a machine and run your programs there. So that is how it started. It is the sharing of resources to reduce the per-user cost. So that was the idea why servers came into the picture. They were their shared resources. And of course, now this sharing leads to reduced cost per unit of computing or per unit of per user. So this lower cost was essentially the driving force behind having servers all across.

(Refer Slide Time: 11:44)



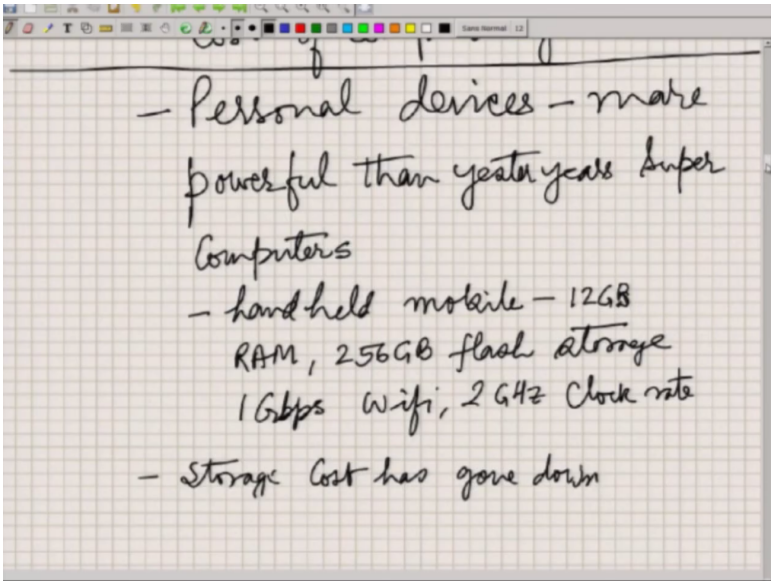
And so now this becomes kind of a de facto thing, so everybody has been talking about servers only and talking about data centres, and people will access services from there. But things have changed over time.

(Refer Slide Time: 11:57)



So this good business principle of sharing costly resource so that utilization become very high and which will reduce the cost has to change now. Technology has changed, and the cost of computing has gone down drastically.

(Refer Slide Time: 12:15)

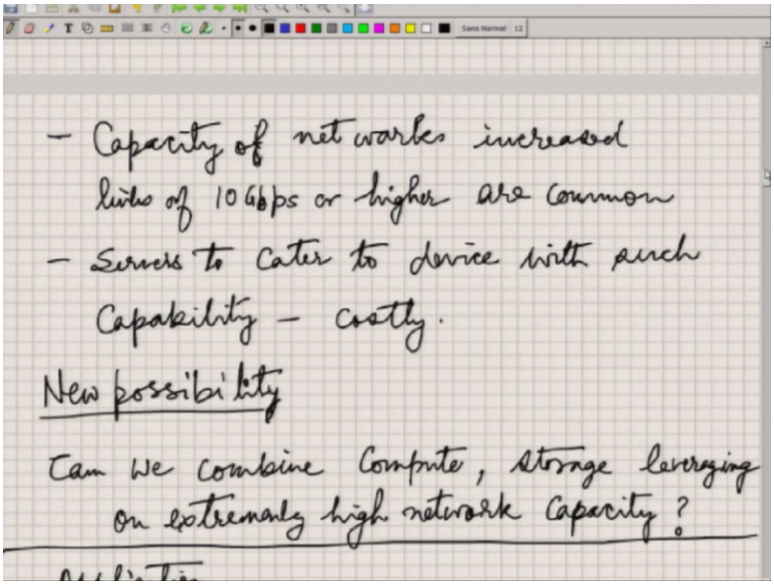


So now even look at personal devices, the situation with them, and the more powerful they used to be there in the past. So even, in fact, some 25 years back, if you look at a supercomputer now your mobile phone, which is octa-core running at more than two or three gigahertz, having 12

GB of RAM probably that mobile phone is more powerful than that supercomputer. So handheld mobile phones with 12 GB of RAM, 256 GB of flash storage, one Gbps Wi-Fi, two gigahertz clock rates, or even higher is very common now. So which such a kind of powerful thing and they act as a client, you need very high capacity servers. And of course, you will not be able to fully use these devices unless servers of extremely high capacities are being built. Or alternately, each server will be able to handle now less number of users.

Now, we also have a drastic change; the storage cost has gone down. You can see 256 GB is a huge space, and maybe in time to come, mobile phones with one terabyte kind of space flash storage will be viable. So this actually means now much computing is available and resources are available at the user end. So servers probably have to go now.

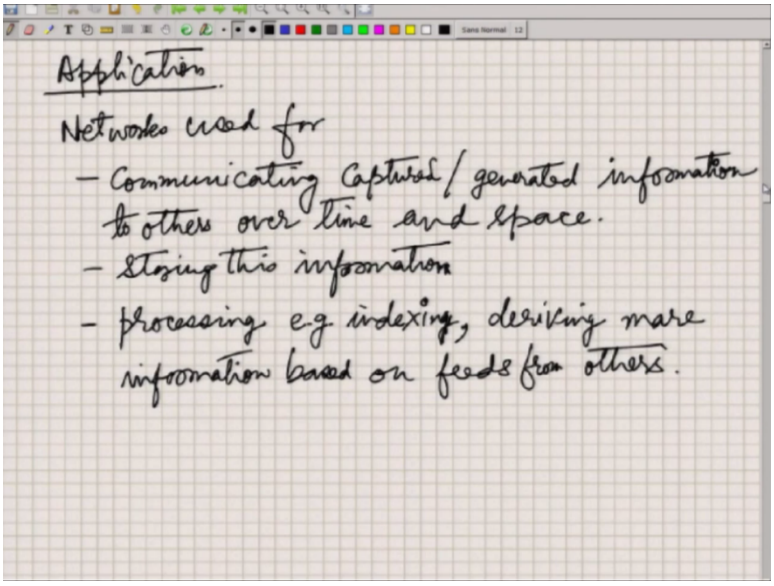
(Refer Slide Time: 13:37)



Another thing that has happened is that networks' capacity has increased drastically because optical fibre has been inducted. And now, you can easily think of 10 Gbps or higher links. IIT Kanpur has two 10 Gbps links, which we cannot have imagined when joining this profession here. And when you want to build up servers to cater to these devices, either the number of users per server has to go down, or server has to be also, their capacity has to increase their compute, storage, and bandwidth test, everything has to go up. So this will become more costly. As we cannot combine the compute of the user end devices, their storage, and because we have very high bandwidth available, even to the user, connect all this and network all this capacity to build

up the systems instead of using servers. So we can, can we do away with the servers? That is a question. And certainly, it should be feasible, and that is what the Peer to Peer networks.

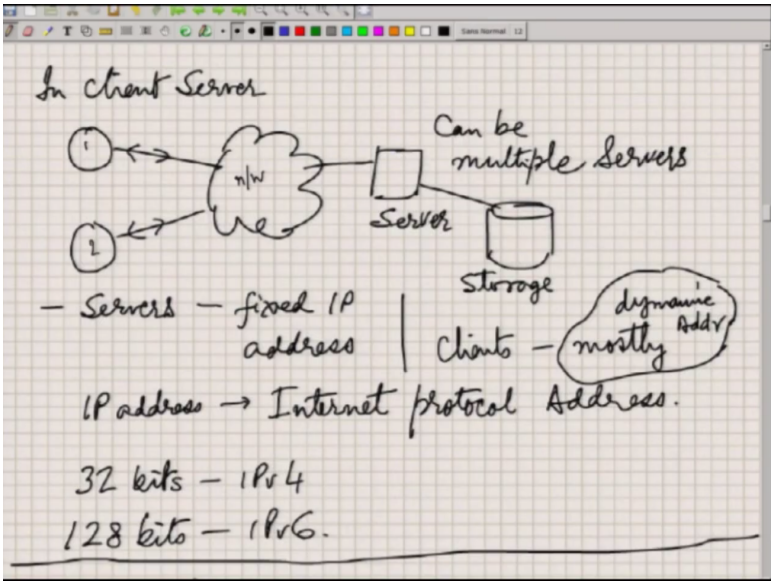
(Refer Slide Time: 14:49)



Now normally, what kind of applications are existing on the Internet, let us look at that. So the networks, which are, our Internet is being used for communicating the captured or generated information. You are doing Facebook; you are getting information, or you are generating information and pushing it to the network, and other people are accessing it. So this is not to over space, not it is not immediate; it is also over time. So you do it now today, somebody else gets it tomorrow. So, immediately only over space, telephony is a good solution if you want to do it; it is a live thing. So alive as well as live as well as non-live kind of communications, both are supported.

You also store the information that implies. You store it for archival purposes; you store it for later on access. And once you start the storing, you also can index the information, and you can derive more information based on the indexing or processing. So all this functionality has to go into the server. All these capabilities that happen because of the network and servers have to be now can. Can this be done? When I build up user machines being assembled to build up a distributed server less system, we essentially call it Peer to Peer system.

(Refer Slide Time: 16:15)



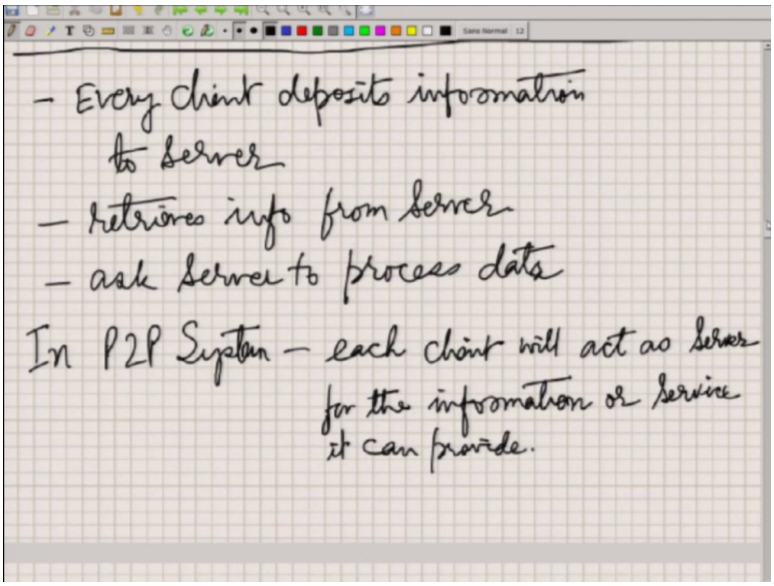
Now, there are problems when I talk about this, so let us look at what specific issues we will get. In a client-server system, we have to address each of these issues when designing a Peer to Peer system, so I have specifically mentioned those things. Here you can see in a client-server two nodes are one and two here are being put. The client-server model is connected to the Internet and connected to the server. And these servers can only not be one. They can be multiple clusters of servers, but they are only visible as one machine for the user.

Now, these servers need to have a fixed IP address. Every machine on the Internet for routing the packets is a packet switch system Internet sending information to him; we need an IP address if you are following TCP IP network. So these are going to have a fixed IP address. That is one important thing. But while you are using a mobile phone, so your location changes, its IP address keeps changing dynamically because this IP address is being allocated through a server, a small DHCP server in the various zones of a telecom operator.

So clients mostly are going to have dynamic addresses, while servers are mostly based on static addresses. That is the current way of handling the things, and these server addresses can be IPv4 can be IPv6 both 32 and 128 bits both. Now when you change your location because of mobility, your IP address has to change so that the packets can be routed to you. Different zones will be there because we need routing aggregation. I do not need to maintain routing tables for every device.

For a zone, I will keep one routing table entry at other places. So all addresses have to be somehow related. So when you move into a zone, your address has to change. And that is where the dynamism of IP addresses does come for the clients because clients are mobile, servers are not supposed to be mobile, they are static, and they are in one place. So they can have fixed IP addresses.

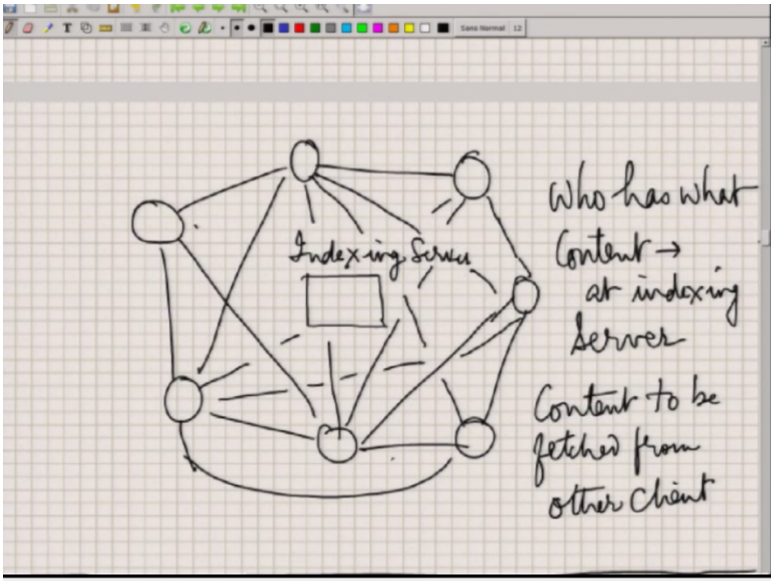
(Refer Slide Time: 18:28)



Maybe one option you can have is if the clients have to work as a server to tell about themselves to continue to the mechanisms that will identify where the server is; I will talk about that. Normally, what each client does as of now is it will be depositing the information to the server. As you are doing it to Facebook, when you are running a Facebook client, you are retrieving the server's information, you are asking it to process some data.

What do you want to know to make the changes is that there are no servers? You want the information you fetch it directly from the other clients, other users, and you want to give the information or give it to other users. You want to process it or ask some of your clients to process it. But the problem is other clients, their mobile, their endpoint addresses will keep changing, or IP addresses will keep changing. They may not be on because you turn off your mobile phone; you can turn off your laptop when you do not use it. The servers' reliability is missing, and the IP address's static nature is missing and is still we want to do the function. So let us see how it happens through indexing servers.

(Refer Slide Time: 19:49)



Usually, the idea here will be that we need to maintain something like this if we somehow maintain something called an indexing server. That is a naturally occurring way of solving this problem. Every user every peered client, for example, here I have shared all peer clients can talk to each of them, I have not shown all the links. So there are seven of them. So it will be ($^7C_2$), okay.

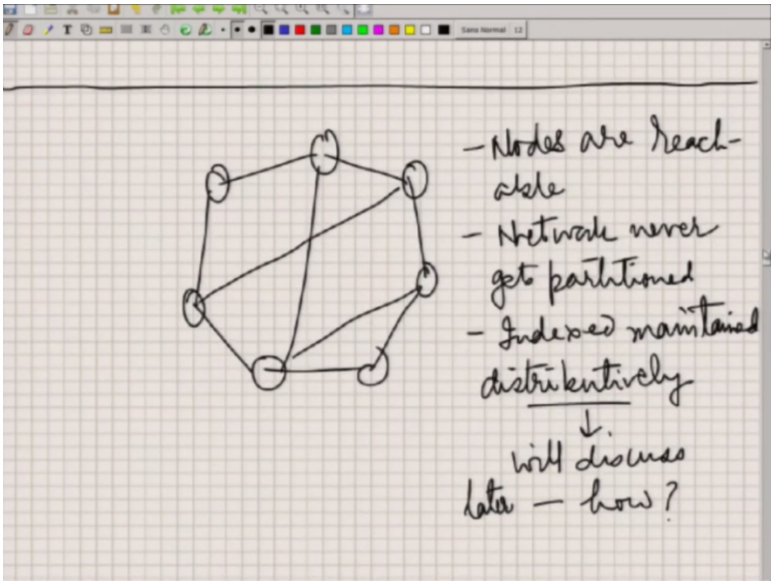
(7*6)/2 so, 21 links will be there among them to connect to each one of them. So is the overlaid network so they can over the Internet can directly communicate with each other. So that is a virtual link. Now we need also to maintain a server. Each of them will publish in the indexing server what kind of resources it has, and its IP address. When somebody wants to search for a resource, it goes to the indexing server, and the indexing server will search through the resource and find out who has this particular resource, IP address, and port number? And then you can directly make a connection and fetch the resource. So, the indexing server is a central repository through which you can identify who has what resources.

We even do it now. You want to, for example, find out something about wanting to learn about some content. For instance, you want to find out, what is a coronavirus, what you do? You normally go to www.google.com and put in coronavirus in the search engine. Consequently, you will get many websites, which are mostly static because they are fixed IP addresses assigned to them, this is not the user client machines, but it is an index. So www.google.com maintains an

index. So it creates a search index by crawling through all the static web pages all across the world, static and dynamic, both kinds of contents.

And once you get this thing, you can fetch from that server. Instead of the client, you are fetching it from the server, and these are mostly static addresses. When you want to build up this kind of index, you require dynamic addresses to be handled.

(Refer Slide Time: 22:09)



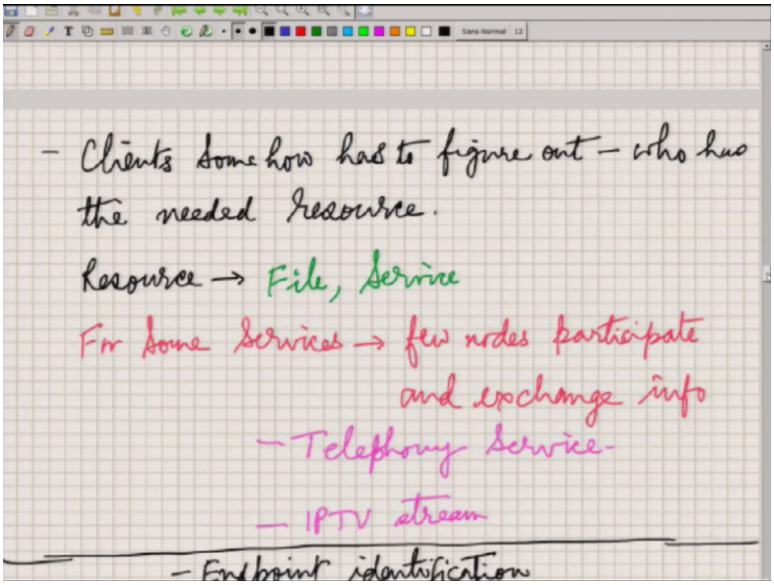
And one essential thing because nodes are acting as a server and client in a Peer to Peer system, this network should never get partitioned. There has to be, and we create an overlay technically into this, this should know, I should be able to reach, I should know everybody if I know who my neighbours are and their neighbours. So it knows about the information; it is not maintaining connections is reachability.

So, reachability to every node in the peer network should be maintained. So the network should never get partitioned so that one half of the network does not know even a single node from the other side. Again these two cannot reach each other that should never happen. So that requires a kind of a trick. I will explain how that will be done when we talk about Peer to Peer routing tables.

So these all can now maintain this index the centralized index, which I talked about earlier. This all now have to be managed in a distributed fashion. Can this be done? So, not only this

distributed fashion in which I am going to store all the index entries will allow me to maintain the entries, but also it will give me a method by which the network will never get partition that can also be guaranteed; this is what we call Logarithmic Partitioning.

(Refer Slide Time: 23:40)

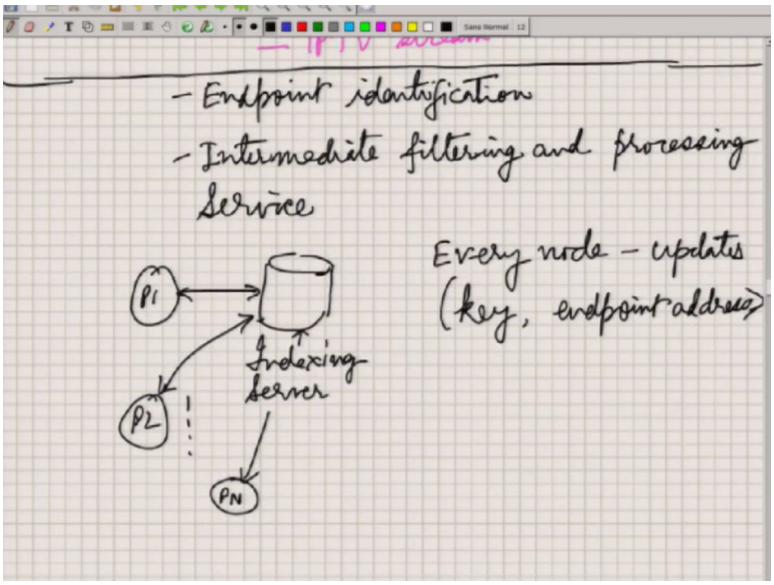


Each client needs to find out, figure out somehow that who has the resource. So now, what could be this resource? Let us talk about this resource. So this can be a file, so that is what you do with the webpage. You are trying to search for Coronavirus things, who have published the files about coronavirus. So you go to Google, search into the index, get the URLs and click on that and directly fetch the file from the other servers.

There can be services also. For example, you want to make a telephone call to somebody. So this what happens in the SIP telephony Session Initiation Protocol? So all modern-day telephony is based on that. So you normally, every phone will get registered to the SIP server, so the SIP server maintains an index of which user IDs map to which IP address and port number now and keeps changing every few minutes. So as your IP address changes, your record will also keep on changing. When you want to make a phone call, you go to the SIP server, find out the IP address and port number of your destination to whom you want to call, fetch it, and set up a call through that SIP server.

You can find similar to IPTV streaming; it is a live thing service. So the service also can be searched and can be connected. For example, you want to do a compute; you can go to this thing and find out the computing service where it is available, connect to that service and do the compute. So resources can be files as well as services both.

(Refer Slide Time: 25:14)

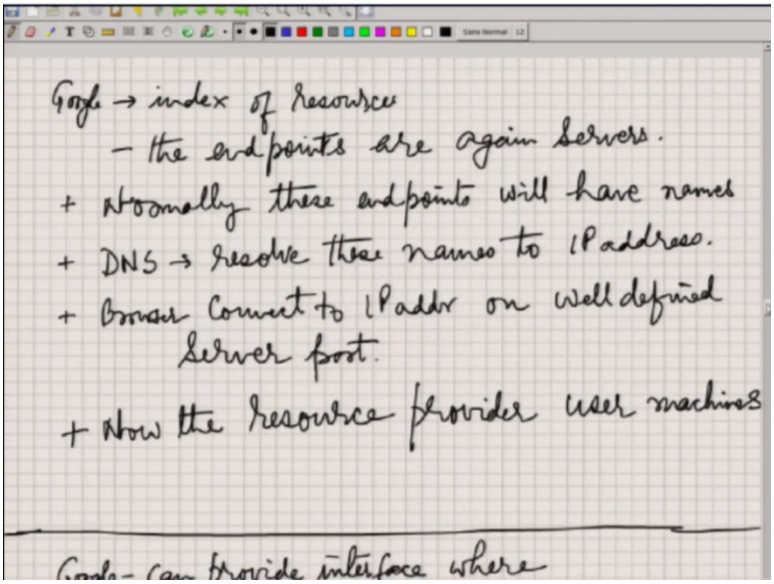


An important thing is endpoint identification. So that is the crux, whether doing it through an indexing server or a distributed mechanism. And if it is going to be kind of a distributed system where all mobile clients act as a server, they need to tell every time the key with which they need to be searched. For example, I am providing a file on coronavirus. Whenever my IP address changes, I should tell the indexing server that updates my address to this. This is a key for which I am holding a document.

The key will be the keywords by which you search the document and end up getting my address. The key and endpoint address is the search key and the corresponding value of that key. So this has to be updated by all the mobile clients so that addresses can be changed. With the static addresses, this is not that much of a problem and is typically a Google, which runs crawlers and searches the web to maintain the index. It was not the applications or the user end, which was updating the index. But now the situation is different from peer clients. So we can have this dynamic index update.

So we have on a similar concept dynamic DNS. So your name remains the same, but your name to IP address mapping has to change. So as your client machines, IP address keeps on changing, you keep on changing your DNS entry. But the entry which is maintained in Google is based on your DNS name, so they do not do it on Google. Google always keeps the resource with the domain name. So it will tell coronavirus.iitk.ac.in will be the name. So name to IP address mapping, you can change it Dynamic DNS; that is what can be used now; many small businesses use this to maintain their websites.

(Refer Slide Time: 27:08)

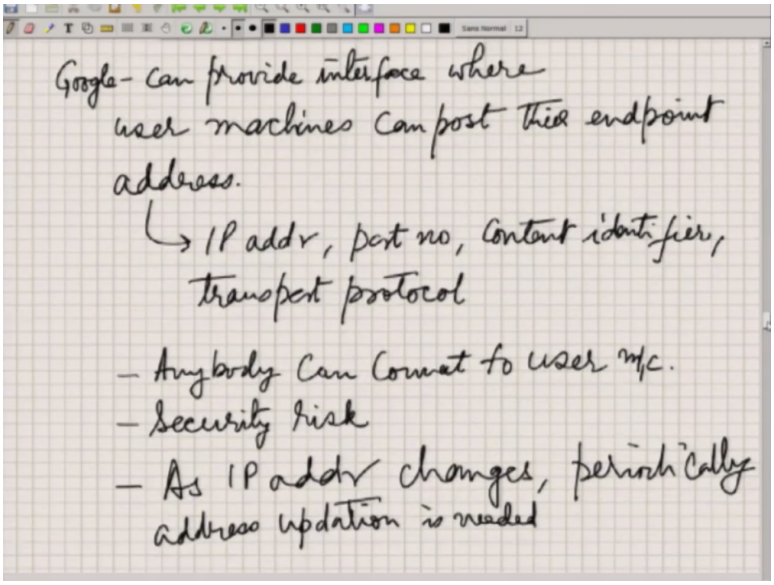


Now there is one important thing in Peer to Peer systems user clients; in current-day thing, its endpoints are server. And as I mentioned just now that they will always be identified by names, not by IP address. There is also a reason why this has to be done because people cannot remember IP addresses, names you can always remember. You know that you have to go to the www server of IITK; you always say www.iitk.ac.in. You do not have to remember the IP address. And I can always keep on changing the IP address.

And you connect to the site; the browser will connect to the IP address on a well-defined server port. So these also have been identified. So for web service, port number 80 is used for the secure web. It is 443 for proxy caches 8080; many ports have been defined. And only change now we are making is now resource providers will be the user machines. So we have to also do

away with these well-known identified port numbers. So port numbers can be dynamically changed, and we have also to take care of the index in a distributed fashion.

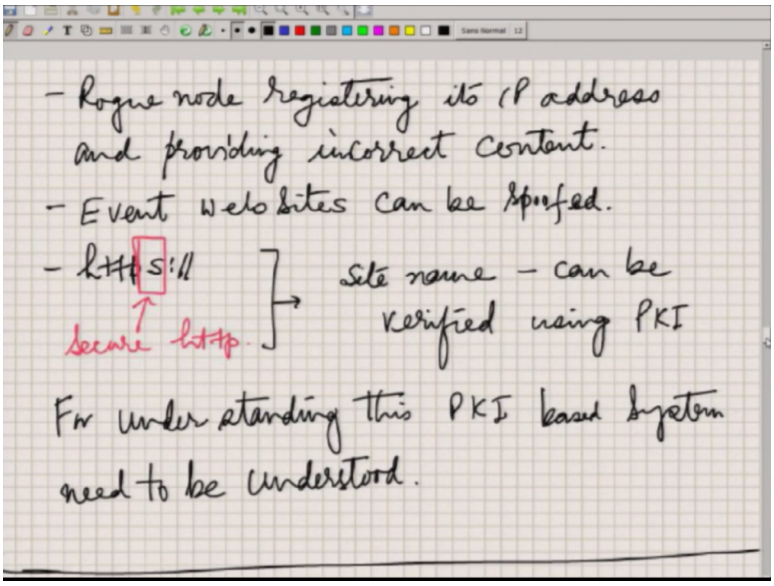
(Refer Slide Time: 28:25)



As I told, Google can provide dynamic updates to update it dynamically port number and IP address both the content identifier and the transport protocol. We will just change this to a distributed system later on. This is a centralized indexing. So this way, now, anybody can connect to the user machine and then fetch the data. But here, one of the problems is how you will be sure that a user machine is an authentic machine, is a genuine user.

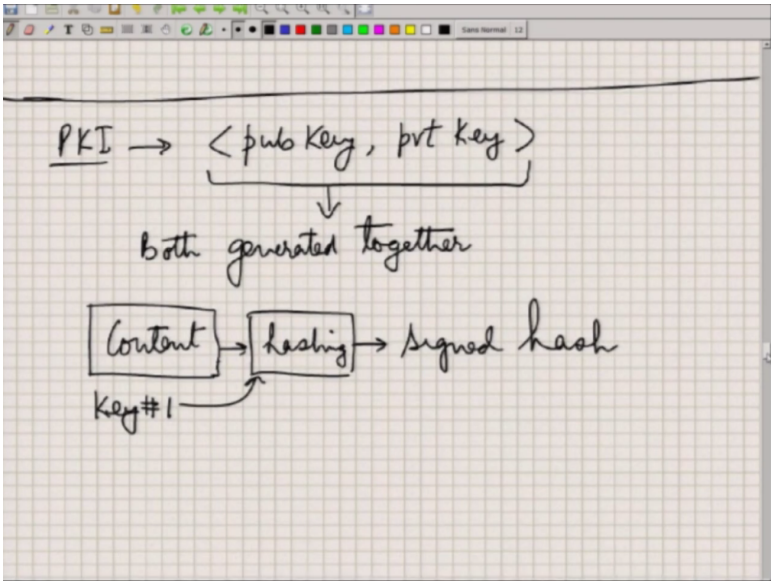
On servers, this is done in a certain specific question using security certificates. And if the risk with this content with the user machines is that as IP addresses change periodically, they need to be updated. So but you have to verify this guy is genuine even when the IP address is changed, somebody else is not faking as that guy. So the security issue has to be handled.

(Refer Slide Time: 29:30)



The problem is how to take care of when some rogue nodes can register its IP address and provide incorrect content. So even the websites can be spoofed, but website spoofing now has been more or less resolved by forcing the browsers to use HTTPS protocol. Otherwise, there is a warning which has been shown to you. Now, this S stands for secure, secure HTTP. So you can verify the site team whether it is genuinely the owner of that site name is that machine or not can be done, this verification can be done. So we need to understand public-key cryptography for this. So this we need to understand. So, let us move to that.

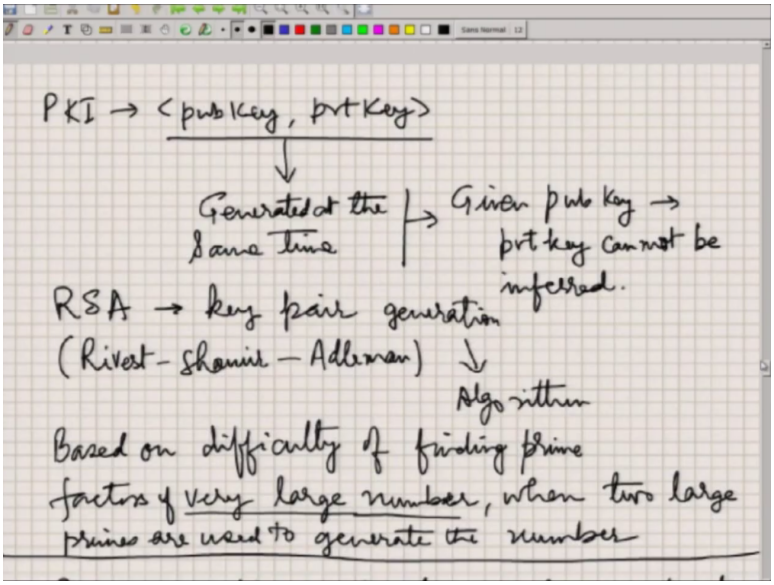
(Refer Slide Time: 30:18)



What is this PKI? So, this essentially means that we can generate a public key and private key, and the good thing is that both have to be generated together. You cannot generate one, and you cannot know the public key and find out from their private key or cannot know the private key and then find out the public key, which is not feasible. So both have to come out at the same time from the algorithm.

One of the popular algorithms is the RSA algorithm for doing this. And there is now even a better variant; the cryptography-based system is two keys that can be generated. One important thing is that if I have a key one, for example, I take the private key, I take this content, and I use a hashing function and is basically keyed hashing. I can give key as an input, content as an input, a hash that gets computed, and the hash is a fixed-length string. If I change my key, my hash will change. If I change my content, then also hash will change. So the same content signed by different keys will lead to a different signed hash.

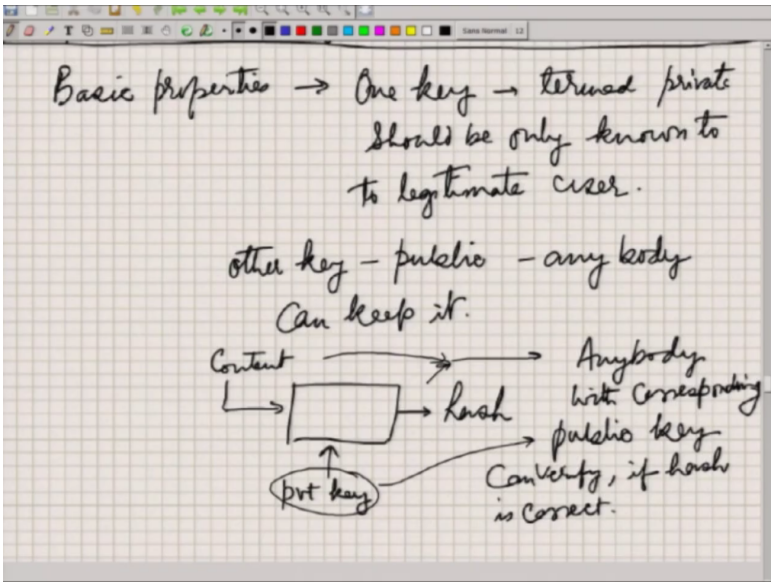
(Refer Slide Time: 31:37)



So this mechanism actually can be used. Now, as I said that two keys have to be done together and knowing one, you cannot find the other. That is a key thing. Now, these key pairs can be private, and another one can become a public key. It does not matter how you do it, but both are like pairs. And the algorithm came with Rivest, Shamir and Adleman from MIT; actually, they gave this algorithm key pair generation. And the problem here is you take two large prime numbers, make a product of them, and get a number.

Now doing the reverse, finding out the prime factors is difficult if you know the large number. That is the basis that finding out two large prime factors of a large number is difficult; that is what is being used to create this whole system essentially.

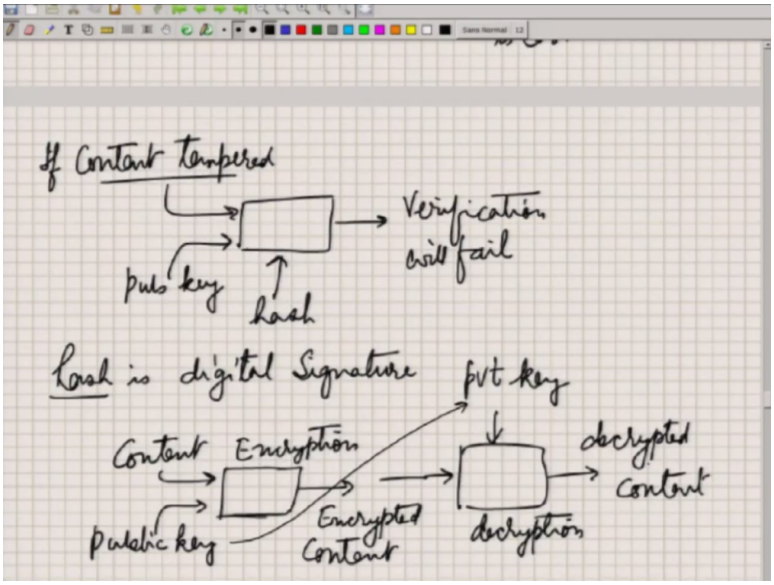
(Refer Slide Time: 32:30)



Now, this is the basic property. As I told that one key would be used as a private key, it should only be known to the legitimate user who owns the key. Another key is public. So the other public key will be known to everybody. So this guy will tell my name is so and so, this is my public key. So this public key is, anybody can keep it. Now what you do is if I want to publish any content, take the content, use my private key okay, and then generate a hash, and this hash will be published along with the content. So, these two things will be put together and will be published to the users.

Now users will have the corresponding private key. So there is a corresponding public key. This public key can be used to verify whether this hash was indeed generated by the corresponding private key in the content. So if the hash match does not happen, then there is tampering in the content, or the hash must have been done during the transit. Therefore nobody can tamper with the content, and we can do tamper-proofing of the content with a signature process.

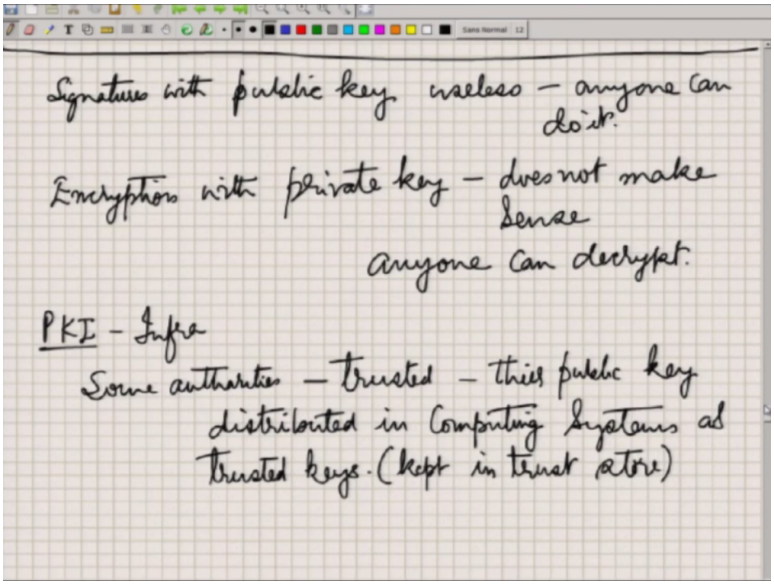
(Refer Slide Time: 33:49)



Now, this is where I have shown it. If your content has tampered with, you use your public key, input compute the hash, and put in the hash, the verification will fail unless this public key corresponds to that private key and content is intact. So even a single bit has been modified, the verification is going to fail. This is a digital signature mechanism.

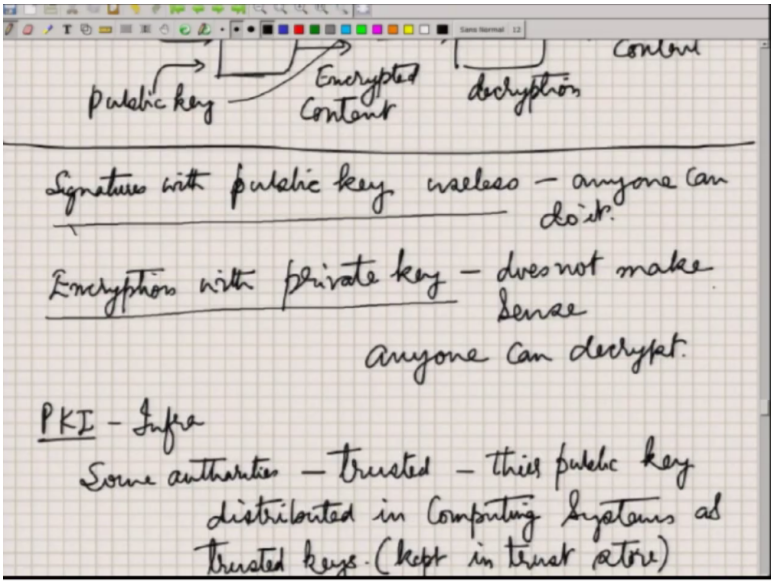
We can also use the same thing for doing encryption. So you take content, insert the public key, run through an encryption algorithm, and get encrypted content. When received by the user, this encrypted content has the corresponding private key. He has to use this private key this encrypted content and get the decrypted content back so original content will be recovered. Now you need to understand that nobody except the guy who holds a private key can decrypt this. So this is pretty secure. It is asymmetric; the public key is known to everybody. So everybody can send encrypted information to one single guy so nobody else can read it.

(Refer Slide Time: 35:06)



These two things in combination can be used to build up our system this HTTPS, which I was talking about, verification system for a website. We have to use something similar in Peer to Peer system. Now one thing you have to ask, we should actually ask, anybody who is logical will always ask when I am doing signatures here, why I am not using a public key. So when I want to generate a hash, why not public? Now, if you can do it, anybody can generate the hash. So it does not make sense.

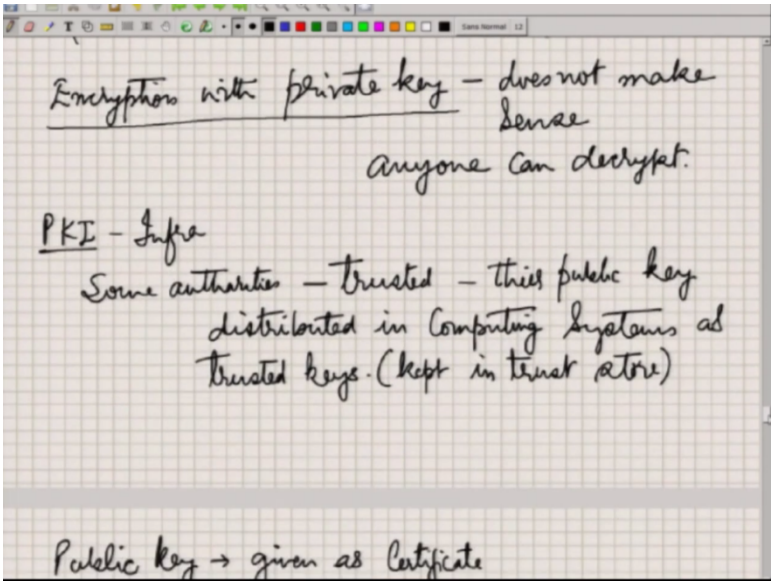
(Refer Slide Time: 35:39)



If signatures are done with a public key, they are useless; anybody can do their signatures. Signatures should only be doable by the owner, nobody else. So it has to be done through a private key because he is the sole person knowing the private key. Similarly, doing encryption with a private key does not make sense because anybody who holds a public key known to everybody in the world can decrypt the information. So there is no use in doing that encryption if anybody can read it. So encryption is always done with a public key, and the private key will always do signatures. Other keys will always be used for the reverse process.

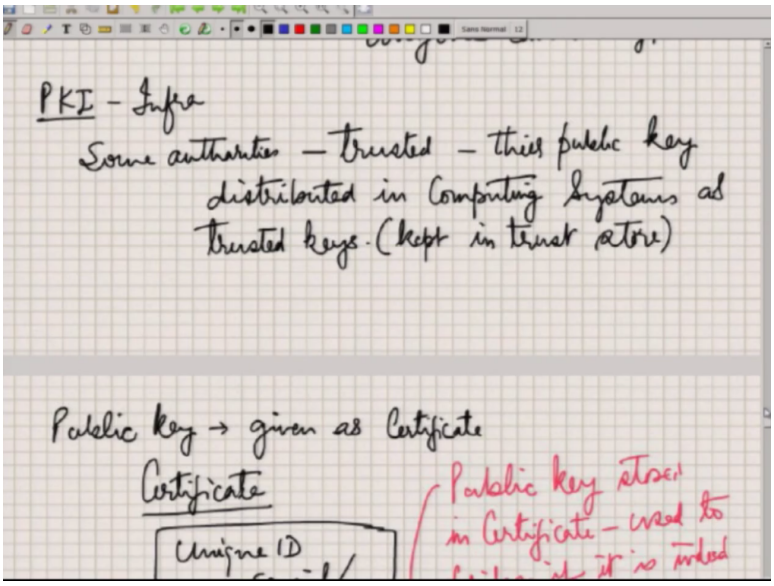
So normally, the way it is done is that we have to create infrastructure. So we will have something called trusted authorities, certification authorities, and their public keys will be well known to everybody. So if you look into a browser, you can go into settings, passwords and security. Inside this, there will be trusted certificates. So, where I think every few months, there will be a browser update or operating system update, and these certificates will get updated. So it is the trusted public keys on which you have a trust. So these are certification authorities.

(Refer Slide Time: 37:04)



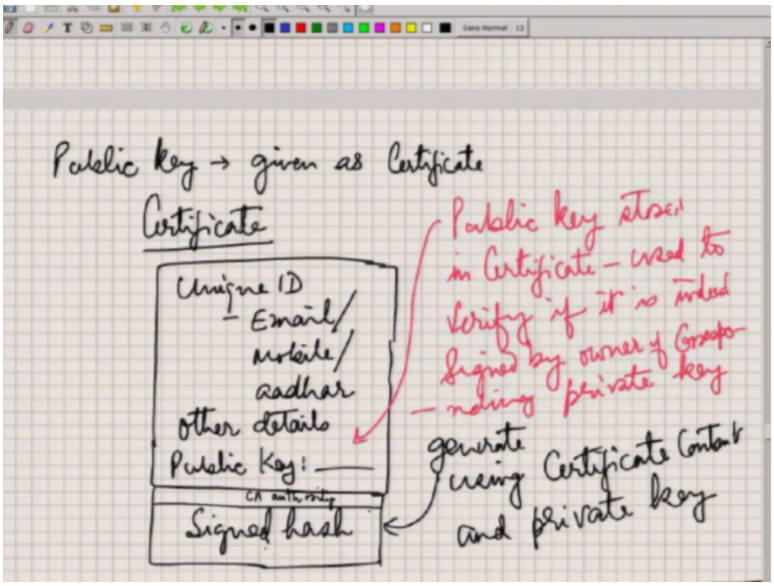
These are distributed periodically, and these are used to essentially check if the other end is genuine or not genuine.

(Refer Slide Time: 37:09)



And these are kept in a trust store, we call it trusted public keys, and we call it a trust store where the trusted information is kept.

(Refer Slide Time: 37:22)



Now the public key is normally given as a certificate, so it will have a unique ID email, mobile, Aadhar, or other details. You can put the public key there, and this information can now be signed. And this signed hash can be generated by a certification authority, so if there can be another field here, which will mention that who is the certification authority here. So this CA authority will use his private key to generate the signed hash. So, now anybody who will look into this will look at the CA authority, signed hash, it will then look into its own trusted store

whether CA authority exists there. And that CA authority certificate will be a public key that will be there in a trusted store. It will use that public key to verify the signed hash.

And once the signed hash has been verified, the guy is genuine. So all this information is fine. The public key, which is mentioned, is also fine. Therefore, the public key stored in the certificate is used to verify if the corresponding private key owner signs it. So we can have extra information if you wish in the certificate, which can be self-signatures, which can become part of the certificate itself. That depends on how you configure it, but typically, the self-signatures are not required.

Whenever you generate a public and private key, you will be creating only this information, you're unique ID, email ID mobile number, everything and your public key. This information you will be giving it to the certification authority. Which, in turn, will do one thing, it will verify every information provided by you. For example, verification can be done using OTP on your email id or mobile number or maybe Aadhar number.

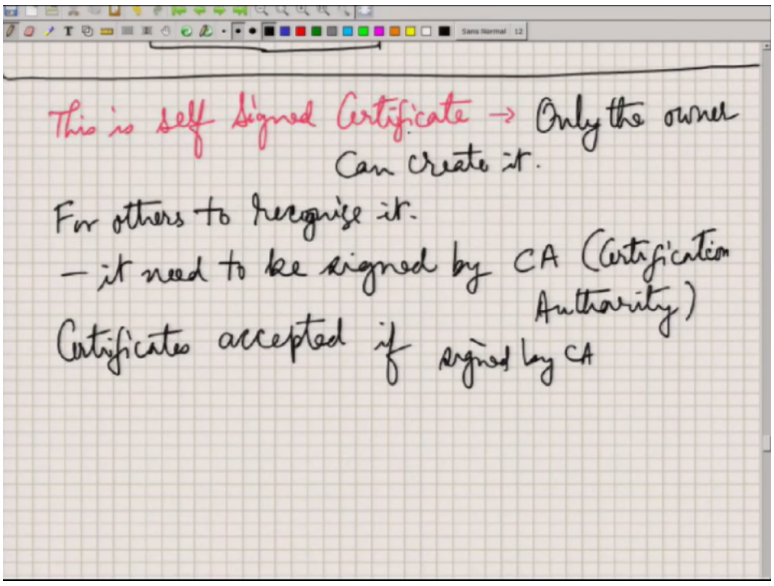
While that is being done, OTP, so this information is verified, you are the owner of that email ID, which will verify that you have the private key or not for the public key you have submitted. So it can generate a random number, which then can be encrypted with this public key can be sent back, and the client can decrypt it back and send it back through the public key by encrypting again through the public key of the server. The certificate authority will then get everything intact back, so the guy who submitted the request owns the private key and has access to the email ID or whatever other credentials there.

So it will now digitally sign the certificate by its private key and send the certificate back. Now, this is very important. So servers also keep the certificates with them. This is generally the method used, but we can do it the other way around, where a certification authority further signs a self-signed certificate. But that will ensure that extra step of verifying whether the guy who has submitted the certificate generation request holds the private key or not, that has to that step can be avoided because that is automatic gets added to the system.

So in our design, when because this whole thing which I am teaching is based on a development project which has been going on here in IIT Kanpur, I call it Brihaspati 4, which is a Peer to Peer system, not only it will provide LMS but other facilities also. So we have already done it apart; it

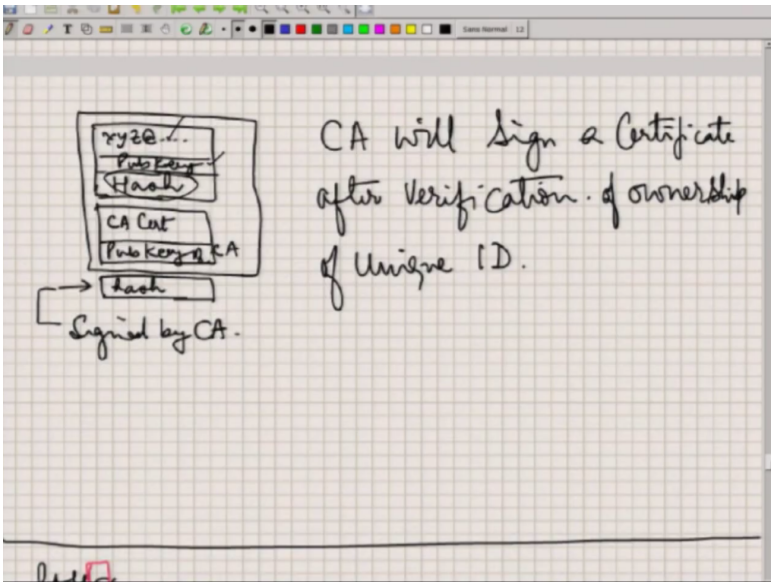
is an open-source project on GitHub, so I am teaching from there. So we have done the design of the certificate in a slightly different fashion, but it is done differently in real life, but this is still flexible.

(Refer Slide Time: 41:13)



This is what I was talking about. If it is a self-signed certificate, the only owner could have created it, so you do not need to verify that. And when CA signs it, it means this was a genuine guy email, and everything has been verified by CA authority, and CAs public key would have been there in a trusted store of all the clients so they can accept it.

(Refer Slide Time: 41:38)

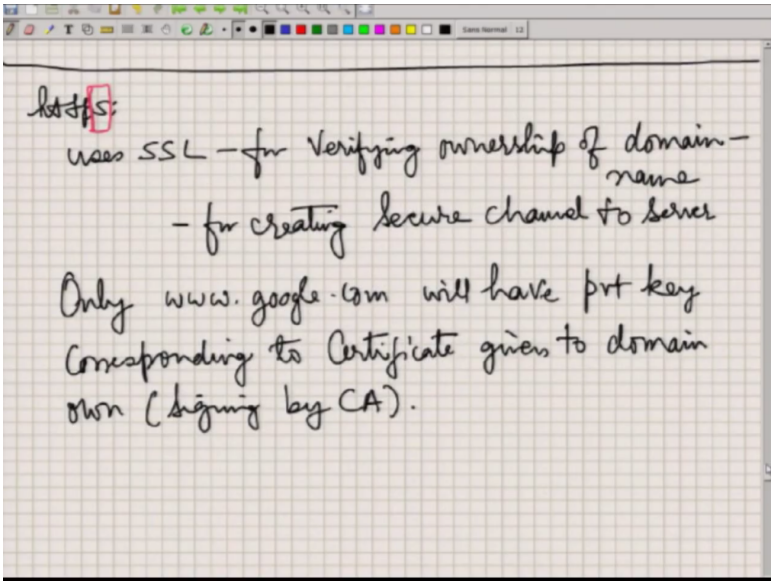


Normally, this is the format that we have been designed for Brihaspati 4. However, this hash, this part of the hash will only not be there if I am going to use the conventional structure, but we have kept it in our structure to avoid one more step. So this is how the certificate will look like.

We are now able to verify the ownership of this email ID, we can verify whether this corresponding private key of this public key is available with the user or not, and then this hash and this public key CA thing is available in the trust store of others, so they can always now have faith on this if the public key of CA which they have in their trusted store, through which they can verify this and figure out the hash.

Once they figure out the hash is justified is verified, then this all is genuine. So, then they can figure out talk to the SSL, talk essentially a small mechanism by which they can figure out whether there is a genuineness of the other end can be found. So let me go through it; you will appreciate this one.

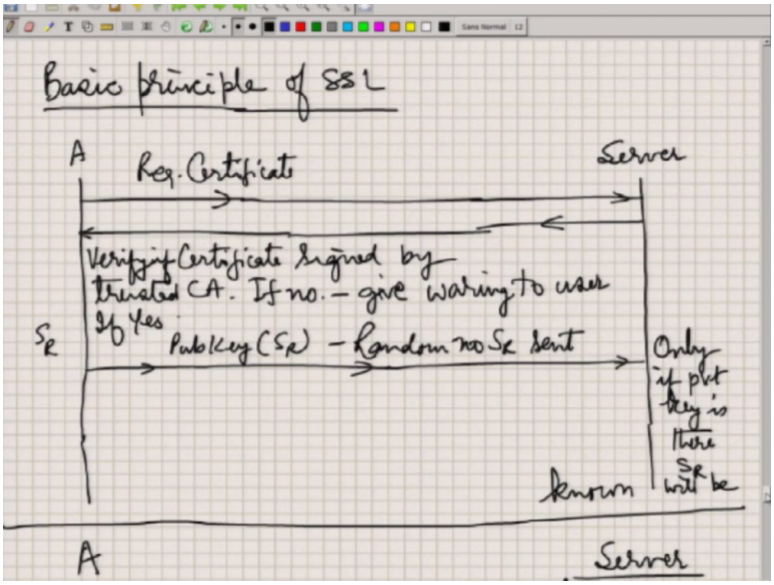
(Refer Slide Time: 42:54)



Now, this comes to the S part. So how this certificate will be used to verify the server end when you do HTTPS. Now S stands for secure socket layer SSL, which is used to verify the ownership of the domain name. And then, you also use it for creating a secure channel to the servers. So, S is used to verify the server's identity and create a secure channel.

For example, www.google.com will have a private key corresponding to a certificate given to the domain owner. So domain owner is again Google, and this certificate must have been signed by the CA Certification Authority, whose public key must be there in your trust store in your browser.

(Refer Slide Time: 43:50)

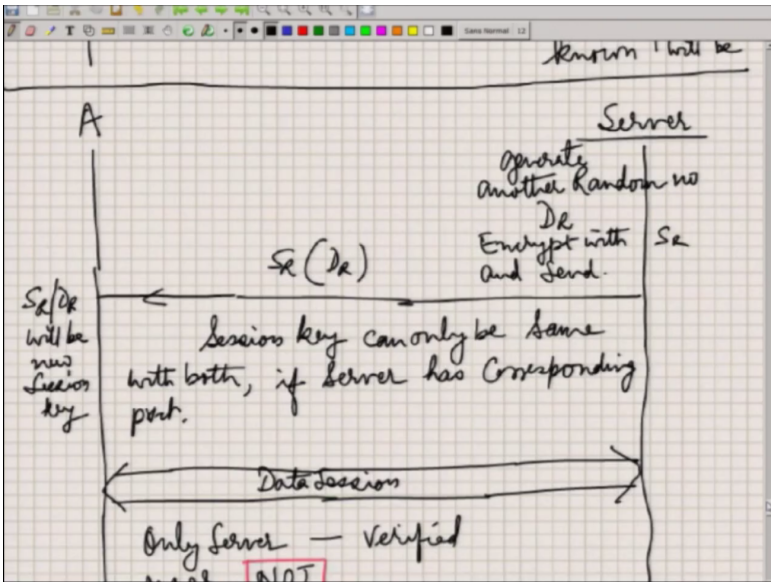


The process will go something like this. This is an asymmetric one. So when your browser, which is A here I call it, is the SSL principle, it is not exactly the SSL or secure socket layer. So A will talk to the server it will request for the certificate. So the server will return the whole certificate intact, and certificate A will verify a trusted CA indeed signs the certificate. It will warn the user if it finds out a trusted CA does not sign it or revoked. There is a revocation mechanism also.

So, and if it is yes verified certificate is fine. Now it has to; user A has to find out if the server owns the corresponding private key or not. So just presenting, because anybody can present you the certificate, but the important thing is a corresponding private key is being held by the server or not. So for that, A will generate a random number I call it as SR. It will be encrypted with the public key.

Remember, if the server holds the private key, it can only decrypt back the SR; otherwise, it cannot. So this random number is going to be sent, SR will be sent back to encryption. Only if the private keys available here SR can be decrypted; otherwise, it cannot be.

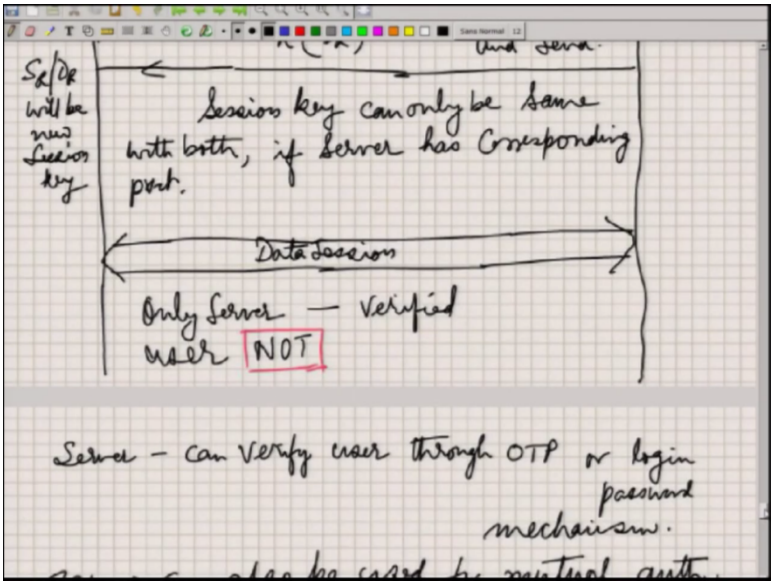
(Refer Slide Time: 45:11)



Once a server is genuine, then only the step will move ahead; otherwise, it will not generate another random number called DR, it will encrypt with the SR, and it will send it back. Now SR is not being sent in this reverse path. In this path, it is not being sent. So, this way. SR is only used for encryption purposes; DR is being sent. So once the DR comes back, it will decrypt it with SR and generate a new session key SR DR.

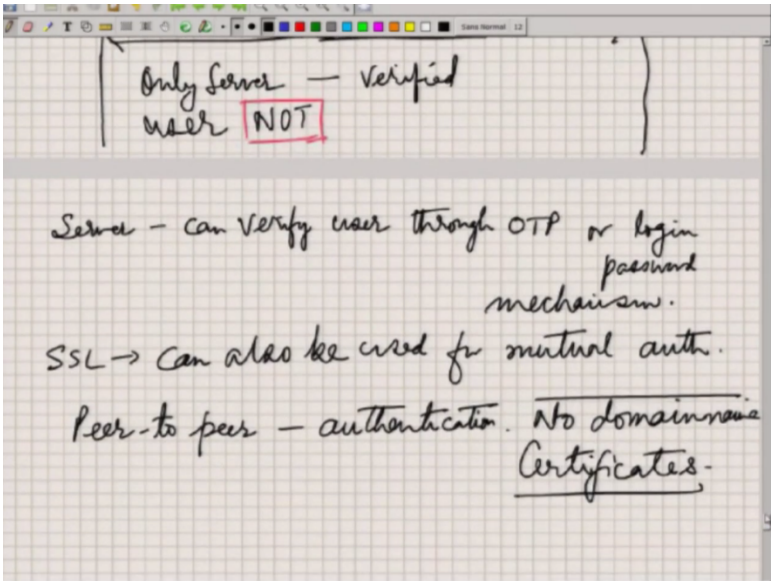
Now, on both sides, I need to generate random numbers so that even if you are using the same random number, other guys may not be using the same random number, and you will have a different session key. So you cannot listen to earlier communication cannot guess the new session keys which will be generated is for that purpose. So session key will be computed here. The server will do the same. And they will now do a secure communication over this using this session key.

(Refer Slide Time: 46:04)



So not only have you verified the server, but you have also now created a secure session. But note here that the server is not verifying A, A can be anybody, A is not presented certificate, A has not been challenged to prove that he holds a corresponding, A is not even, may not even be having the certificate. That is what happens when you are using a browser to access gmail.com.

(Refer Slide Time: 46:34)

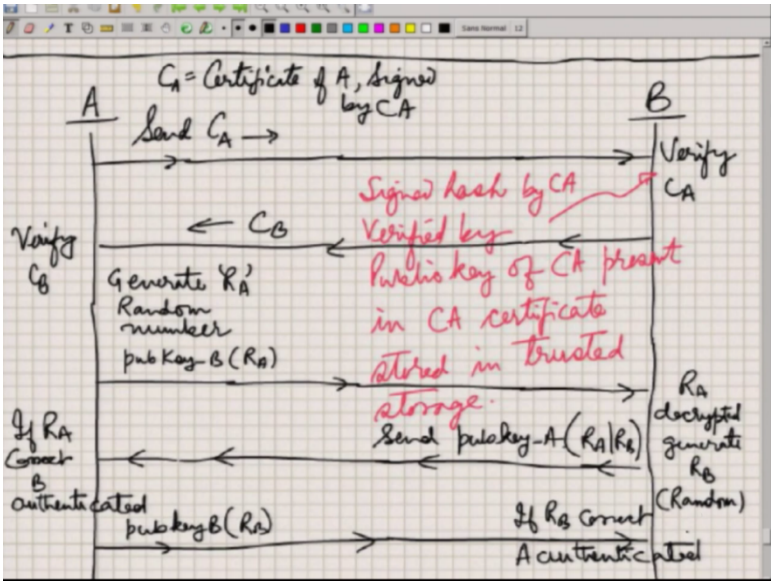


Now what the server can do is use either OTP or use a login password mechanism to verify you, so you do not have to own a certificate. Because when you have to own a certificate, you have to generate a public private key pair, send all your information to somebody, CA authority will

charge some money, who will verify all your credentials, then will issue you a certificate, which may be valid for a year. So that all licenses have been taken care of the common man.

So we can use this same SSL in a slightly modified way for verifying each other, not only the server. So that way, that problem that IP addresses of the users will keep on changing but the user id say email id remains the same. I should still verify the other guy; even when the IP address is changing, there is nothing like HTTPS. So but we can use this; the same certificate can be used for doing verification. For Peer to Peer authentication, we will be using a user identity certificate, not the domain name certificate, while for server authentication, it was the domain name certificate.

(Refer Slide Time: 47:40)



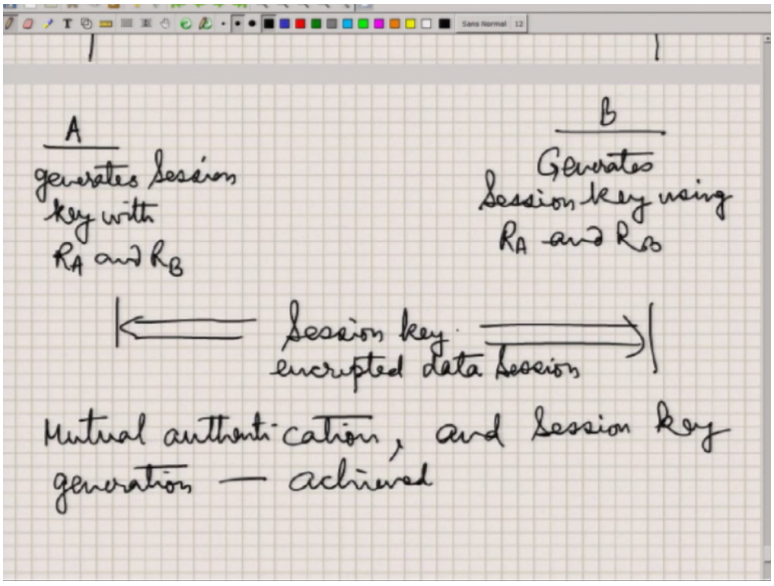
This is the process where two users or two peer clients will do mutual authentication using this. So CA will be the certificate of A signed by a certification authority. When A and B want to talk, A will send a certificate CA to B. B will look into a trusted store and find out the CAs public key will verify the certificate. Once it is verified perfect, it will send back their certificate CB to A. CB will, A will also do the same process. It will look into the trusted store, find out the public key of the certification authority. From there, it will check whether the hash is correct or not for the certificate. If it is, indeed, it was issued by CA, so CB is correct.

Once these two are correct, then they can start the mutual authentication process. So what they have to do is they have to verify whether the other side does hold the corresponding private key for the public key, which is present in the certificate. So that is a test which has to be conducted.

So what A will do is will generate a random number RA, it will encrypt that random number with the public key of B and will send it to the RA, send it to B. B should be able to decrypt this RA if it holds the private key for this corresponding to this public key. Once it is done, it knows RB and RA; it will now generate another random number called RB. This is like a nonce; nonce is a randomized component so that things do not repeat.

So RB will again send RA and RB concatenated will be sent through public key after encrypting through the public key of A node A and this will be sent back here. So if RA and RB both will be, you will be able to decrypt back. If RA is being recovered correctly, which has been sent, you have verified that this guy was the original genuine guy and sent back RB available. So, because of the correct RA, B is authenticated. RB has to be sent back with B's public key. B will be recovered, so B is also genuinely received correctly received RB. So B has authenticated A.

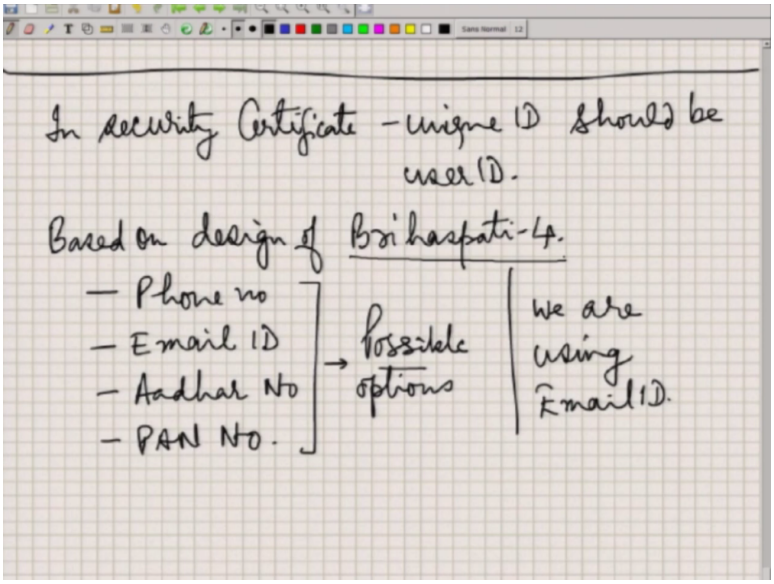
(Refer Slide Time: 50:01)



Both A and B will use RA and RB and generate a session key, B will also do the same algorithm, so there should be the same session key. Now, they can talk securely with each other. They both have mutually authenticated, and they both have created a secure session. Now there is no domain name in the certificate. Now it is an email ID or a mobile phone number between two

peer clients and good enough for mutual authentication. So IP addresses only for sending the packets, it is no more use for authentication, a domain name is not required, and there is a primary method of mutual authentication between two peer clients.

(Refer Slide Time: 50:43)



Throughout this lecture, I have talked about certificates, ultimately for mutual authentication and the unique ID, which have to mention because that will identify each peer client. As I said, it can be any one of these 4. It is for our design. Depending upon what you are implementing, you can make the changes. For Brihaspati 4 project, we have done these four entities.

Now Skype, for example, uses your email ID for doing this verification, Telegram is using your mobile number, WhatsApp is using your mobile number. Telegram uses you; there is a telegram ID, authentication is not through mobile phone, but they also allow you to generate your unique ID. Tox is using 256 bit. I think a random string, and you can give any pseudonym there. There is no IP address anything, and Tox is also one of the beautiful Peer to Peer systems for messaging, and you remain anonymous there.

So with that, I end this lecture, and we will explore more about Peer to Peer systems in the next lecture.