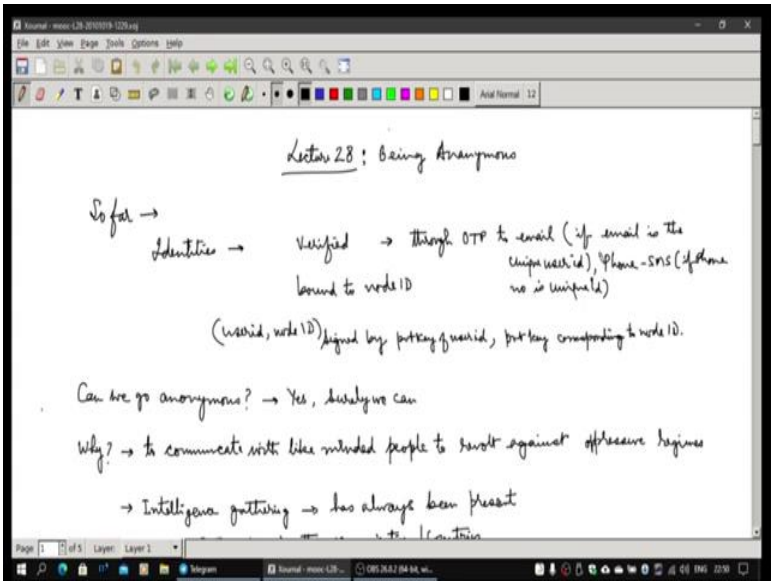
Peer To Peer Networks
Professor Y.N. Singh
Department of Electrical Engineering
Indian Institute of Technology, Kanpur
Lecture 28
P2P Internet: On Being Anonymous

(Refer Slide Time: 00:14)



Welcome to lecture number 28. So in this lecture, we will be talking about how you can be anonymous while in a peer to peer network. This also has been a dream of people who want to be on the internet, but nobody should know who they are and where they are actually at any point in time. So we will look into that.

So far, what we had done earlier was we had looked at identities; we have used identities earlier in all our discussions. So these identities were verified by sending an OTP. So naturally, we have taken identity like an email in our Brihaspati-4 platform, which we have been building. We could have taken any phone number, mobile number, so where SMS could have been sent, and we could have verified.

So, either phone number could have been used as a unique ID or email ID, b4 has been using email ID, and this then gets bound to node ID, so typically node IDs is what is used

in any DHT network for all kind of DHT routing and of course maintaining network connectivity by using a logarithmic partition.
So, and of course, when this user ID node ID binding is done. So, because node ID is a public key or derived from a public key, so there is a corresponding private key, it has to be signed by that. And of course, this user ID node ID combination also has to be signed by the private key, which is corresponding to the user ID; user IDs are also remembered as a public key is an email ID.

And this email ID has been issued a public and private key, and the public key is what is being used for verifications and the private keys used for signatures. When this user ID is publishing the document, this certificate, which user ID has the same user ID to the public-private key pair mapping, is nearly signed by an authentication server, which is done after verifying the OTP.

That is the case which we have used; now the question is, can we go anonymous? Can I avoid using email ID and phone number and still form a peer to peer network? Surely you can do that; in that case, your node ID essentially is your public key. And there is a corresponding private key with the node ID. Now the only problem that remains is how that multiple users, a node can be on the multiple node ID because each node ID will become a separate user.

As far as the system is concerned, though it is possible, it will be slightly tricky to implement; it is not impossible, but currently, let us assume that it will be every node is unique; in that case, what happens? Now the question is, why would we like to go anonymous? Why not show identity? Sometimes, this has been seen around the world; there are oppressive regimes in certain countries.

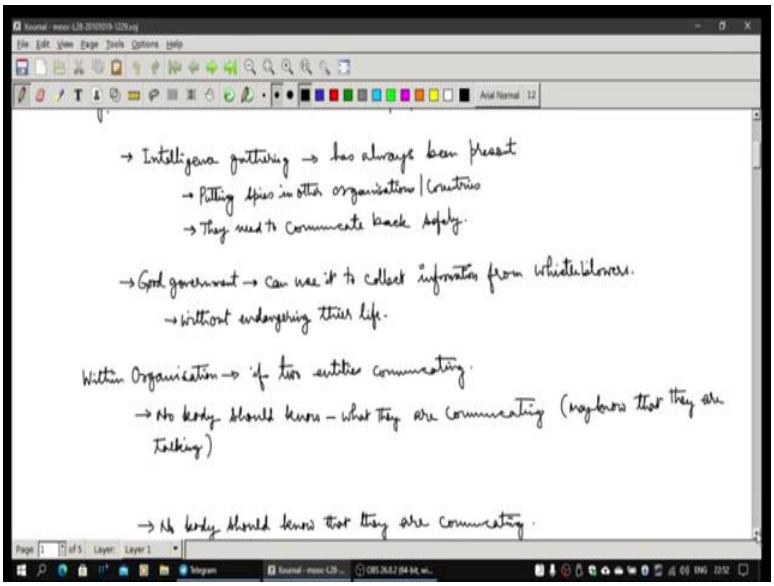
So, people who want to communicate do not want to come into the oppressive regimes' limelight; otherwise, they will be eliminated or killed. So when you want to build up a revolt around that time, this we have seen is happening in Hong Kong people have used

our systems, anonymous systems. Edward Snowden has used them; they have been used to build a Darknet kind of structure.

I think slightly more we will delve into the TOR kind of system later, which is hugely anonymous. And this is one reason why they might be of interest. Then there is always most of the countries, or any government or any leader or any dictator will always be having intelligence gathering mechanisms, so he will be hiring a lot of people almost all government does, and I think they have a lot of I think popular intelligence agencies, we have seen James Bond movies. Hence, MI6, everybody knows about them.

So that is a British intelligence agency, we have RAW, Indian intelligence, ISI Pakistan, Mossad from Israel and so on, CIA, KGB, and then, of course. Similarly, there is a Chinese agency, almost every country invariably, so that is now one can actually gather this information, so what has to be done in that case?

(Refer Slide Time: 04:39)



So typically is unsaid, but this is true that almost everybody plants spies everywhere to collect information. And whenever they are caught naturally, the hostile countries will generally kill them, or they will be eliminated without telling anybody, so it is a

hazardous business. Still, people do get involved in their country. They call it sacrifice to the country.

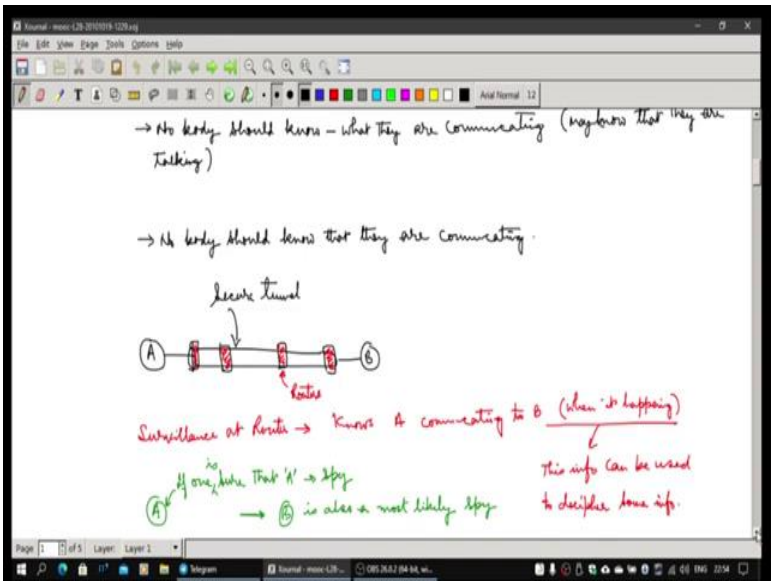
So when they do it, the technology has also been built, so being anonymous also becomes useful in those scenarios. So, because they need to communicate back to their own country without getting into the enemy's hands where the country where there have been planted, this could be one possible way the TOR came out, probably from one such effort.

I am not sure, but my guess is yes, probably most likely it is that. There can be other good uses like good governments can use it to collect information from whistleblowers without endangering their life and improving governance on that basis by eliminating all the corrupt people who are not suitable for the social life, or the whole country.

So, there is also, within an organization if two entities are communicating. They do not want anybody to know who is talking to whom, so nobody should even try these two people are talking, but they are talking, so not only what they are communicating and even that they should know that they are even communicating that also should not become evident.

So, not only what they are communicating, so in those scenarios, this kind of system also become pretty handy, and they evolved because, for that reason. A lot of them are available in open source. I have been mostly gaining how from the open-source community.

(Refer Slide Time: 06:33)

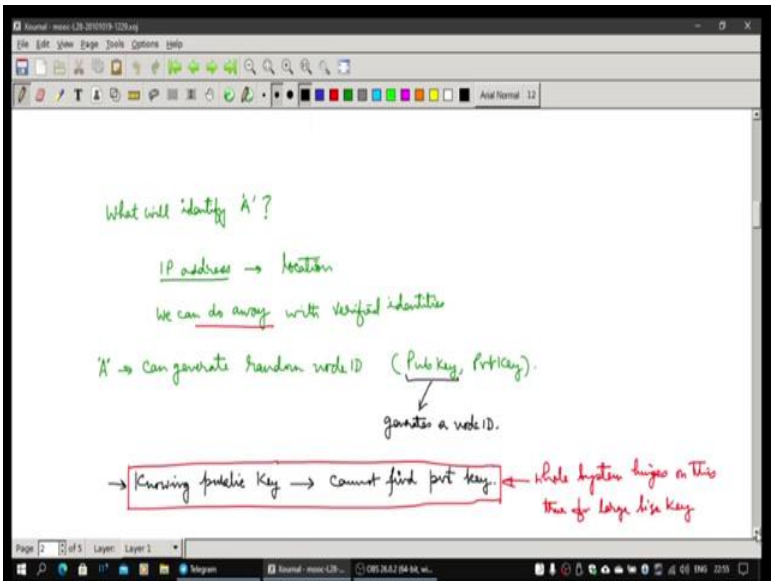


So, for example, A to B, if I want to communicate, what will I do? I will now create a secure tunnel from A to B. Then basically create encrypted things, so it go through routers encrypted packet routers cannot decipher only A, and B has a key, so they know it. But now the problem is routers will know that is A communicating to B, whenever it is happening, that itself is information, so that can decipher at from what time to what time A and B are communicating that itself is an information need to be handled.

And of course, for spies, this becomes complicated. If they communicate somebody from a hostile country back to his own country and the timing is known, if we know they even the IP address what they are communicating is not known so, there is no evidence. But when you do all unethical things, there is nothing like the evidence we do it that time to find out the IP address; they can locate the guy.

We should; we cannot use it. And of course, if you know, the other endpoint is you cannot rely on another endpoint also you cannot rely upon. You can make these connections, and then statistically, you can figure out who the culprits are and then take care of them. So this is a pretty common practice. So, anonymity is becoming extremely important in those scenarios.

(Refer Slide Time: 08:08)

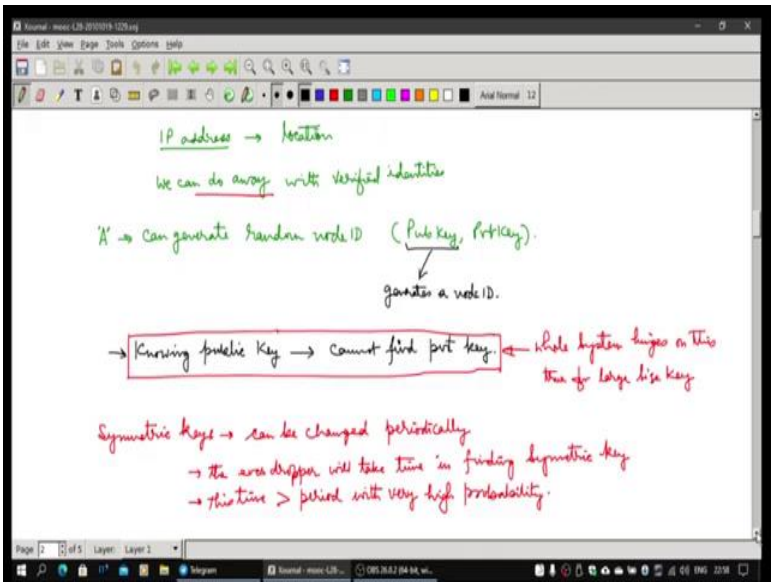


Now usually, the problem here is what will identify A? Its IP address, and from the IP address, you can always get the location; there is no user ID. So, you can do with verified identities, email ID and phone numbers because then it becomes even more straightforward, so those we can throw away. But IP address can still be done, so it is actually not wholly anonymous, but still, yes, the systems are there.

So, I will talk about one such system now. So this is A can generate a random node ID, for example. So his IP address will be known, but this can be taken care of by routing, so we will be doing that; I will explain how it is done. A can generate a random node ID; it will generate a public key and private key pair. And from the public key, it will generate a node ID. So node ID and public key together is essentially is the node ID.

And of course, if you know the public key, you cannot find the private key, and the whole system now hinges on this, which is confirmed when these padded keys are pretty large in RSA.

(Refer Slide Time: 09:23)



It is a Rivest-Shamir-Adleman algorithm, RSA algorithm which is going is used commonly, is pretty much safe used for a lot of commercial transactions. It is called public-key cryptographic infrastructure. But there, this depends on certificates; here, there are no certificates.

So, now what we can do is when these guys communicate, they have there, and they should know each other's public keys. They should be sure it is talked to the same guy, and they can now use a symmetric key for doing all communication. And as we have done in one of the earlier lectures, symmetric keys can be generated through SSL, generating a random number encrypting with the other guy's public key.

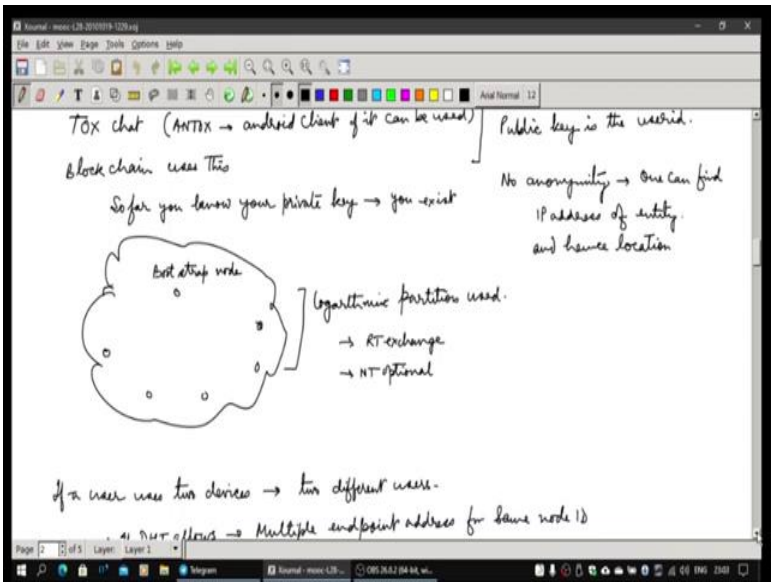
And then, of course, sending it in an encrypted fashion, another guy can get that, he will generate his random ID. Whatever is the decrypted random ID, a random node, the random number, both numbers if you actually can decrypt in the reverse direction, encrypt in the reverse direction using the public key of A, it will come back to public key A and A will get its random number back if it matches so B is genuine, he does have a private key, corresponding to the public key.

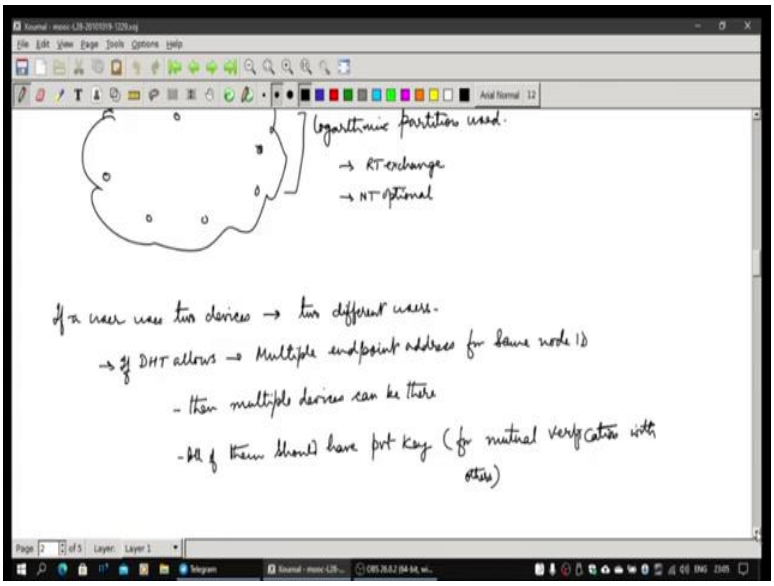
It is A turn to prove that he has a private key, so because he has decrypted, B is a random number. It will send it back again using the public key of the B, and B will decrypt it, and B knows yes, I got from A the correct random number, so A is also genuine. They will now both combine these two random numbers and will generate a symmetric key. So it is not known to anybody, and this can be used for communication.

They now can do something more. This is what exactly is done in some of the products. Intuitively, I can guess that, which is why it is very tough to crack them. So they change the symmetric key pretty fast, every few minutes or even every 5 minutes. If it is done, it is tough to crack because the key would have changed by the time on your crack.

So it will never get the time to find the symmetric key, and if it is long enough, and you are changing it pretty fast periodically. So, you essentially have to ensure that the periodicity has to be good enough. The time to crack the symmetric key is much larger than the period, which happens with a very high probability. So the guy most likely will never be able to crack, and you can keep on communicating secretly. But the only problem is the location can be identified.

(Refer Slide Time: 11:58)





So one of the applications, which is Tox chat, is available on open source. The whole code is available on GitHub, and there is an Android app that is available but not very popular; there is no app, there is nothing. It does not use any server that is one of the beautiful things. And it does message with that, now the messaging here becomes slightly different.

Earlier, when we talked about messaging, we had messaging storage, mail storage we have talked about. So, mail storage is essentially that is what the messaging is, but it becomes slightly different. I use mail storage because the mail was sent between user IDs, not between the node IDs. When I send it using between node IDs, I do not require the storage; the router itself acts as storage because the message is going to be a DHT route.

Now, the public key is the user ID in this case, and of course, there is no anonymity, as I mentioned, so one can find the entity's IP address and hence its location. So this is not anonymous in that sense. Of course, the common man has to get the IP address of the other side first, but since it is going to through multiple hops, unless you know everybody in the hop, you will not be able to figure out what is the IP address.

So, and this can even be made more secure, TOR is essentially based on that, but ANTOX is a pretty simple application, but very useful. You can download it from the play store. It is available for free, and then, of course, try it out. But you would not be getting your friend's name; your friend has to tell his public ID to you, which you then have loaded into your and ANTOX chat and then you can communicate with him.

And you have to tell his public ID that is how he will recognize you; then you can put some name aliases in your mailboxes in ANTOX chat and then communicate with each other. There are no names; there are only public IDs; the keys are known on both sides. Your address book essentially is for each you have to put your aliases or names, but you only remember for all users you will recognize by an alias, but it is a public ID of all users you will be listed in your address.

The blockchain also uses a very similar thing. It does not use any user ID; it uses a public key, basically becoming a user ID. So, any Bitcoins or any smart contracts are assigned to the public key. So if you hold the private key, you hold those individual Bitcoins. If you lose your private key, you lose everything. So far, you know your private key. You exist, you forget of the private key, everything is lost, unless somebody else by chance gets the same private key.

While doing random generation, which is a very, very unlikely, extremely improbable event, so, it will be a loss, it has happened with people, they have lost a large number of Bitcoins that would have been accessible would have been an enormous thing, peoples hard disk has crashed, or the hard disk has gone into the water they could never recover it. If you have lost vast amounts of money, though you search the internet, you can find such cases.

So, the ordinarily important thing is that with ANTOX, there will always be a programmed bootstrap node, so there is at least one node running, which is known to everybody. This comes hardcoded, so when the system runs, it puts in its routing table only one entry by default, a bootstrap node. Since it is there, it starts doing routing table

exchange, so it communicates to the bootstrap node, and the bootstrap node will send its routing table back to you and start populating.

Now you communicate to all these nodes, they will send their routing table. Hence, you keep on optimizing, you will start creating logarithmic partitioning, and Kademlia is a pretty popular thing. I think that is what also has been used in Antox chat. I have not verified this, but that is my intuition because almost everybody has done that. But in Brihaspati-4, we have been using a different one; it is a chord tapestry hybrid.

But we are using assigning user IDs. We have not still built an anonymous system; we have not done that. Maybe in future, we will see if it is required, and then, of course, there is logarithmic partitioning, so the network will always remain connected, as I had talked about in one of the earlier lectures. You can also maintain the neighbour table, but it is optional, but if you do it, you can make the whole network routing far more efficient, and I think we should do that.

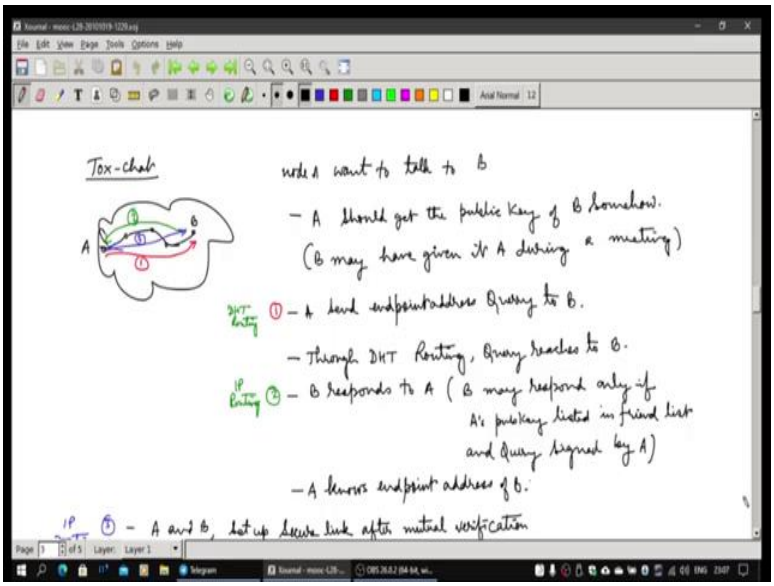
If the user has two devices, there will be two different users because users are identified by the public key, which gives them IDs. So it cannot be the same user. And of course, we can do one more thing, we can use for same node ID multiple endpoint addresses can be given, so in that case, the problem is when you are participating, you cannot participate in DHT; you have to register in DHT. You have two endpoint addresses: node ID, but you are not part of DHT.

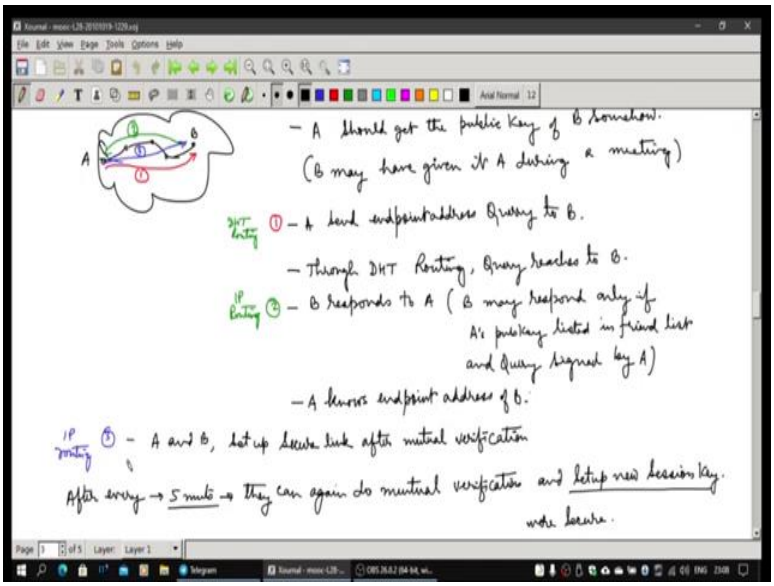
So, if you come to your root note and from there it can be then sent to one of the two. Still, suppose you are participating in DHT; you have to be only one node. You cannot be two because two nodes cannot be placed simultaneously, two different devices. In that case, it will become pretty much confusing; routing tables will have only for each node ID there is not only one endpoint address. There are not two; routing tables do not do that.

So, only as a particular case if when their leaf node it is possible. As told, it is not impossible; it can be done. It is possible to create an alias; there can be two different node IDs, and the both of them we can have assigned information that they are alias to each other, they are the same user for each other. When you send it to one, it automatically knows that the other person I can inform.

So it can now encapsulate the message and send it to other node ID also, so both can keep on synchronizing essentially between them. That can be done but currently, B4, we are not at all investigating this particular thing. When these devices are two devices, two multiple endpoints addresses for same node ID case, so they, when their leaf node they need to both of them need to have the private key for the same node ID, otherwise it's not going to work. They cannot otherwise verify the things, so they cannot do a transaction with any other node. So, how the Tox chat is going to work, let us see.

(Refer Slide Time: 18:25)





So, now let the user A wants to talk to user B. So user A has a node ID A, so I call it node A here, but now the problem is somehow A should know the public key of the B or node ID of B and B should know the node ID of A they cannot communicate.

Maybe when they are met in some way in the conference, they have exchanged their node IDs. So that is what happens through an alternate channel in Tox chat or Antox. These public keys have to be exchanged. And you have to put them in your Tox client or Antox client. So once this is done, what will A will do is; A will now send that endpoint address query to the Bs node ID, and it will be done through a DHT routing.

So ultimately query will reach B, so this is what is being shown here with the red colour, so this actual query goes to B, so now B will respond to A, B will respond only if A's public is listed in the friend list that is what happens in the Tox chat, if it is not there it would not be included in that, you have accepted as a friend then you can do it.

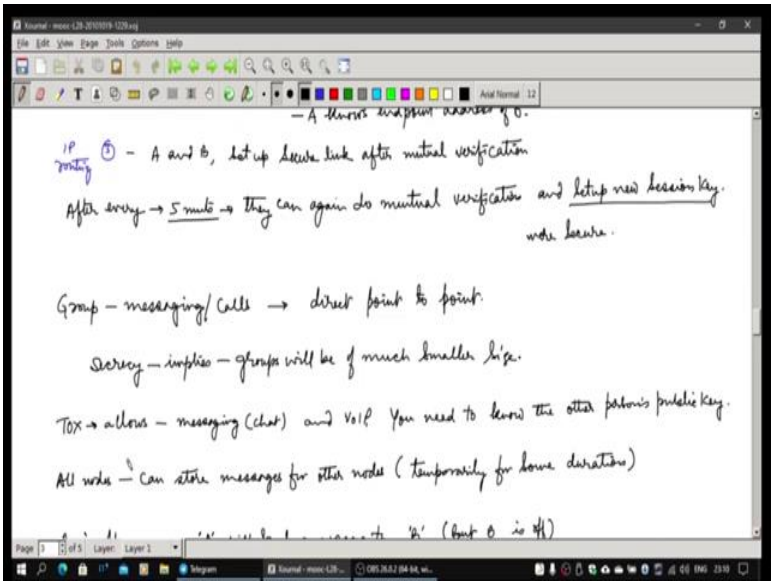
So, B will now also check whether it is saying that A's node ID is this whether it is being signed by private key or not; somebody should not be faking. If the corresponding private key signs it, B verifies, yes, it's coming from the guy who claims his node ID is this, it was signed by the corresponding private key, which can be verified.

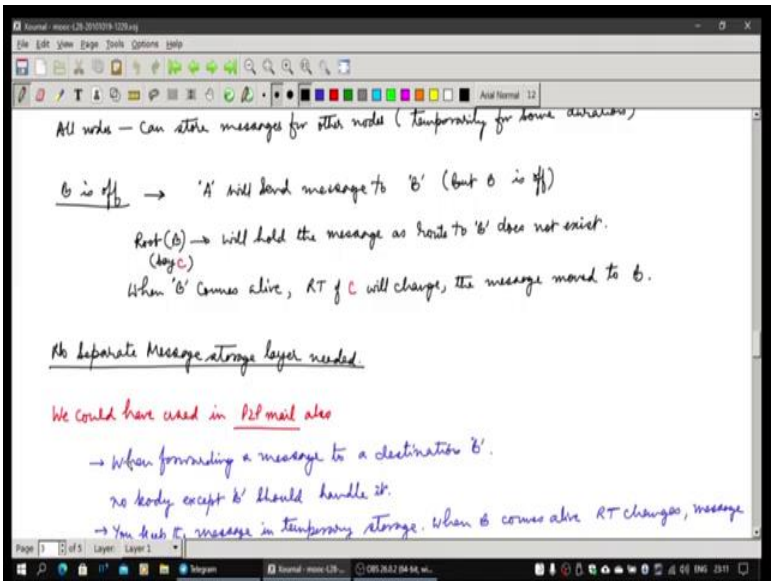
B will send the message back to A in the same fashion, so A can verify it is coming from B. So, A by this time knows the endpoint address of B, and of course, A and B can now communicate through IP routing. They can set up a secure link after mutual verification and transaction; they can make a voice call; that is how the Tox, Antox is doing voice call from between two nodes.

So there is a peer to peer call directly but set up without a server and can make it encrypted. It is encrypted, and the key can be modified every 5 minutes say, so 5 minutes is a pretty short time. By the time you know the traffic through the traffic analysis and find out the key, symmetric key, the key would have changed, so there is going to be an essential, what happens you are working with the older key at the same time you do further another mutual verification and set up a new session key?

So basically, two new random numbers being exchanged. And then you signal to the switchover to the new key, whichever and flip over the other key. The communication changes to the new key, so the key management protocol keeps on happening.

(Refer Slide Time: 21:14)





When you want to create group message or group messaging or group calls, they have to be a direct point to point; there is nothing like this server, the equivalent will not be there, or there will not be any what we call a bridge for the calls, it has to point directly. In such a system, because this is built for secrecy, it remains simple; that is how it is done.

So you usually do point to point call, and that guy will relay information to the next guy. You have to do it one by one, so that is a way for you for the same device you send to generate multiple copies sent to the people. And ordinarily, such systems are going to be for tiny groups, not for huge ones. So the security will be reduced drastically as the number of people in the group grows, so that is a rule.

So the moment to talk about the security number of people in the group usually will be very small. The Tox does allow messaging. It is like chat, it does allow voice over IP, and you need to know the other person's public key that is very important. So, all nodes can store the messages for other nodes, temporarily for some duration, so that is permitted; to be fair, it is not like a message store.

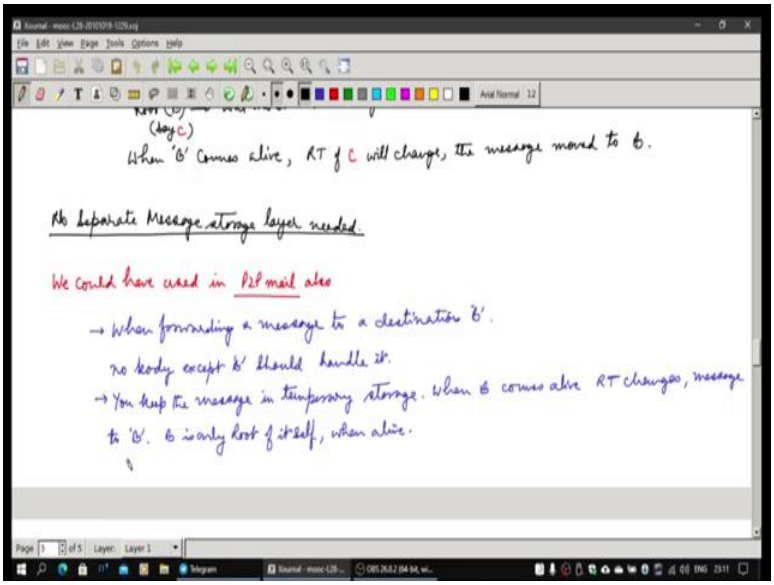
This happens while the message is going through DHT routing itself. The other guy is of destination is off, the intermediate node will not be having that guy in his routing table, a message will keep on lying there because it is the root node for that destination. The only

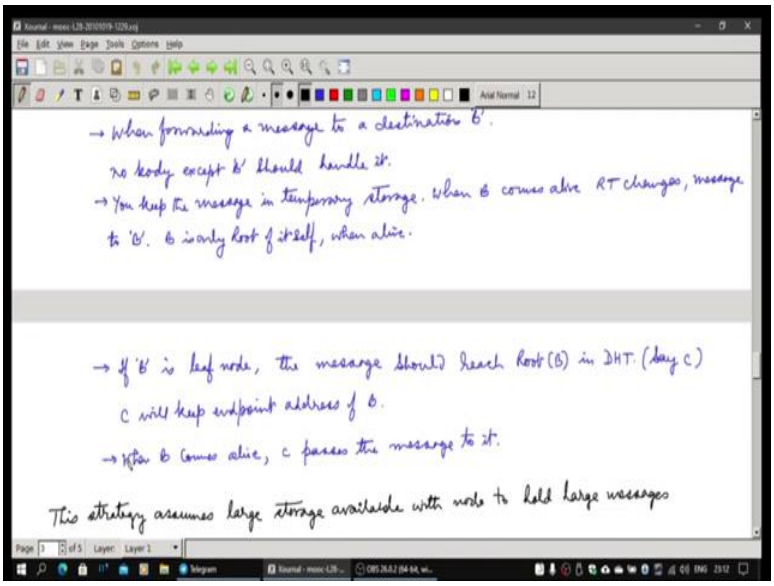
thing the guy who is at the destination when it becomes alive becomes his group node; the message will get transferred and decipher it.

Nobody in between can do it. You always encrypt the message with a symmetric key. That symmetric key is further encrypted with the destination's public key if the message is huge. If it is short, you can directly encrypt with the public key. Only the destination can unlock this coded message and can read through it.

In this case, as I told that root of B, say node C will hold the message as root B does not exist. Whenever B comes alive, the routing table of C will change, and the message will move to B. There is some other better root node for the same message now is going to be there, so there has to be an entry change, routing table change has to happen, this we have discussed earlier also. And we do not require a separate storage layer.

(Refer Slide Time: 23:37)



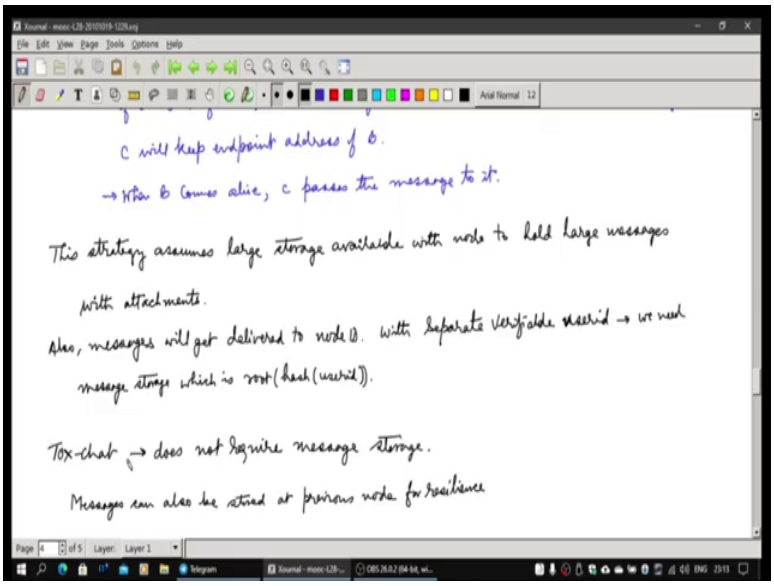


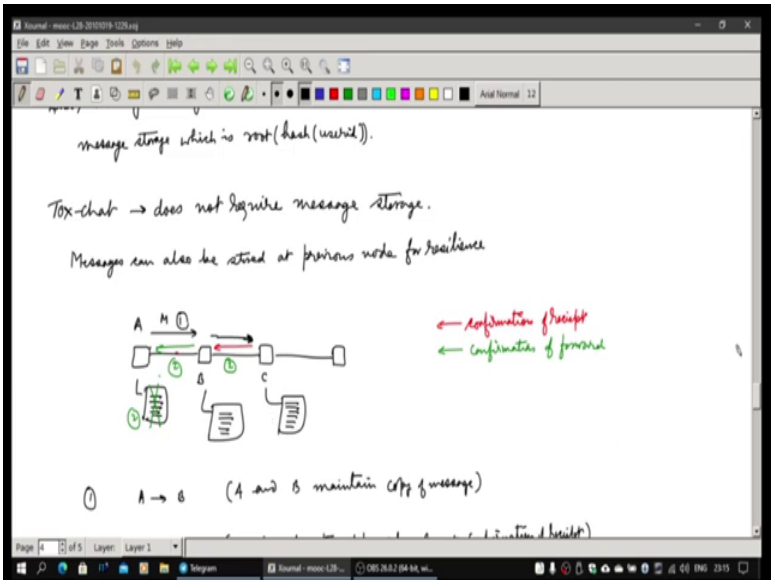
Now, maybe you can question that we could have used this thing in a peer-to-peer mailing system also, no, there you cannot. So, then forwarding a message to a destination B, nobody except B should handle it. That was basically the idea, and we have done this thing here and there also; it is done. It is encrypted.

You keep the message in temporary storage when B comes alive, and the RT table will change. The message will go to B; B is the only root of what I have told for this particular system. Still, in peer to peer, we are actually sending a message, not to the node ID, but the user ID, and it can be on multiple devices registered, so any one of the devices which alive will pick up the message and will then put it in DFS storage which is shared across all the nodes.

Because that is how every device will get the message, but any device can pull-up the message from the message store. We do not have that requirement here, and of course, if B is a leaf node, the message will be reached in the root node, which will be C, and C will have to keep the endpoint address of B. And whenever B comes alive, C will pass the message to it. That is the way it is going to be implemented in the case of ANTOX or Tox chat. In the Tox chat, every node is participating DHT.

(Refer Slide Time: 25:02)





And this means that when you send DHT messages, even internet nodes, you can transmit large-sized messages as an attachment. Again, in a peer-to-peer mail system, attachments can be huge, so we have assumed that if both nodes are alive, they will direct a transaction from point to point via IP connection. They will essentially set up a TCP or UDP connection and transfer the whole message in one go or the attachment size.

Otherwise, they have to use message storage, so the storage size must be larger for a large attachment; it is for a very temporary period until B comes alive and picks up the attachment in the messages box. In this case, probably that is going to be difficult. Tox

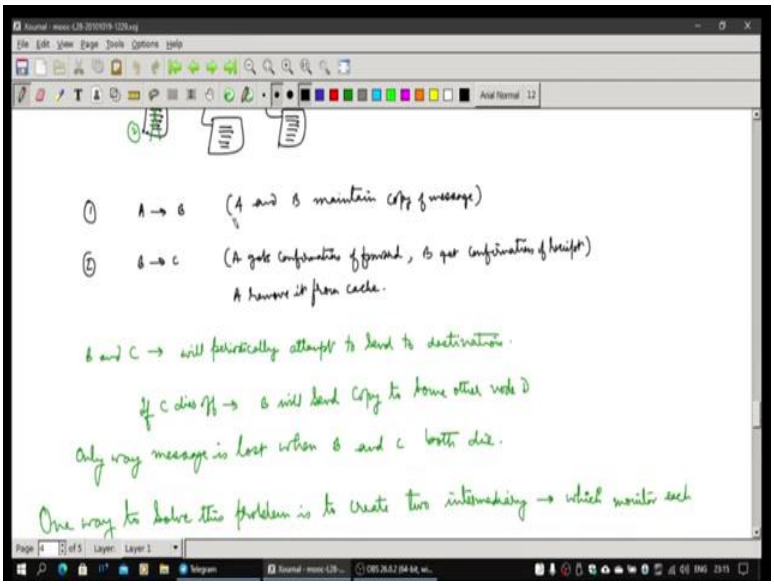
chat does not allow large attachments as of now; you can only send short messages; I have not tested this system very thoroughly. And of course, all the messages will get delivered to the node ID in the scenario.

With a separate verifiable user ID, you require a message, as I mentioned earlier. So Tox chat does not require message storage, and the message has to be stored at the previous node for resilience. Now, there is a problem for resilience actually can come; you can think there is a problem if the node which is holding because B was off that node itself dies off.

For example, A to B to C and then it goes to the last guy, and if C dies off, the message was here; the message is lost. There is no way, so maybe one possibility is that a previous guy can also hold the message. So we can now do this thing that A transmits a message to B, B is not the destination, but it keeps a message with itself, then B has forwarded the message to the next guy, C.

C will say confirmation that I received it. When this guy gets a confirmation, it will say confirmation for the forward, successfully forward so A can remove the message, so there will always be two copies in the pipeline when C transfers it to the destination. It gives a confirmation it has received, C can inform him and remove the message and see once C has got confirmation and it is a destination, it can also remove the message, the message has been delivered. But these are two duplications that can be done for resilience purposes, but there is a better way of doing it. So, let us explain how this will be done.

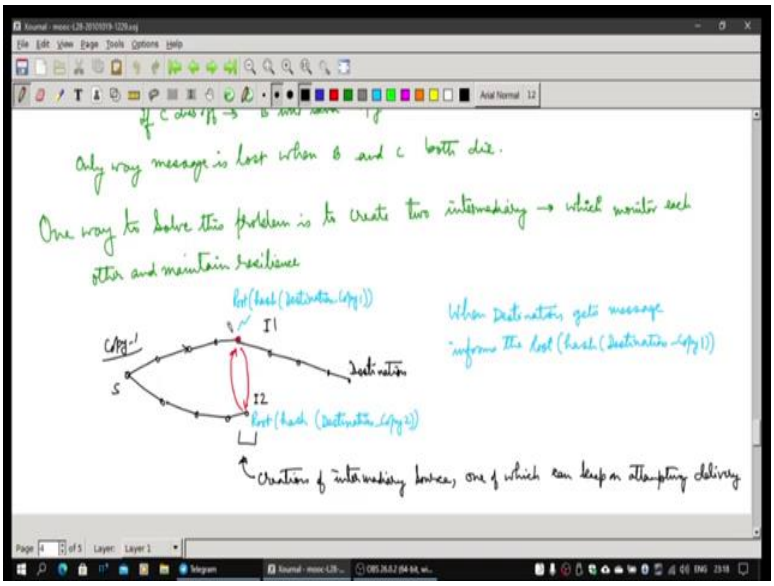
(Refer Slide Time: 27:23)



So, in the case of the same procedure, which has been identified with 1 and 2. So, A sends this one, A sends to B, and both will maintain a copy. B will send to C; A will get the confirmation of the word, it will remove, B will get the confirmation of received. And A will remove it from the cache, only two copies and B and C will periodically keep on attempting to send it to the destination; if C fails, B will route it to some other node, which is now on to the next hop towards a destination and through that hop it will be passing through.

And if B fails, it does not matter C is still making an attempt and it does not there, and some other becomes a better root node; it will try there. But if B fails and then comes back again, you only depend on C; if C fails, it is gone; you cannot do much in this such a small system, but which is okay. You have improved reliability to an extent.

(Refer Slide Time: 28:25)



Another way to solve this particular problem is, as I have to mention, both B and C if they both die, of course, the message is lost, we have improved reliability, but you are not 100 percent sure that message will exist. One way to solve a problem is that you create intermediaries. So, from the source, when you want to send a message to the destination, you now send a copy of one for the destination.

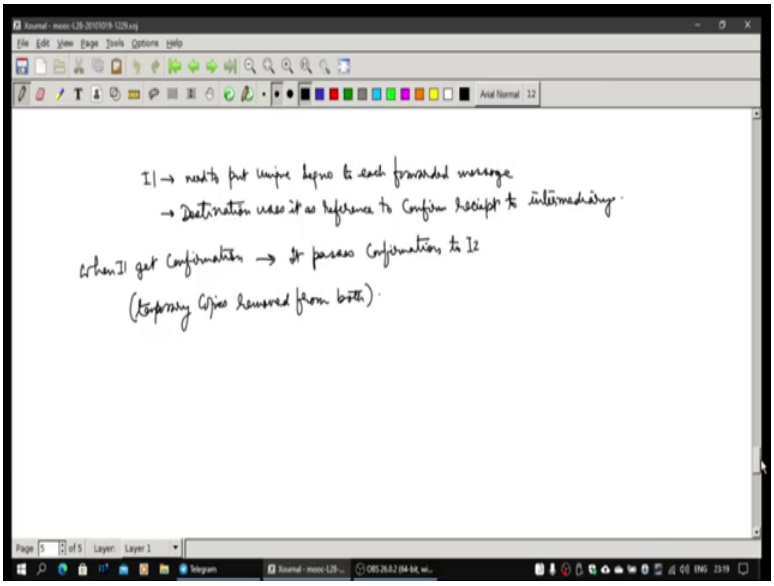
So please do not send it to the destination directly, send it to an intermediary which is going to be identified by destination, append the message copy one string to this compute the hash, and whoever route of that, that become intermediary one. Send the message to him; this guy will not do the copy two to the destination. Create intermediary two, so these two now will start backing up each other if this guy dies off, this will create another copy, this guy dies off, and another node will take over. So there are always two nodes doing it.

This guy may now send a message to the destination, and if there is time out, so destination has to confirm back to him now that I have got the message, so this can then release itself. This can release that particular message, remember. This destination will always remain intermediaries so that all messages will be rotated through this way. But this intermediary always exists; they will never be dying off.

And there is no message store because of a DHT routing. So, first of all, root it here, and you are tunneling it. And from there, then it goes to the destination, so it remains there in the DHT mechanism. So, whenever the routing table there, it will keep on doing it, but it will keep it here till you get a confirmation from the destination. So, that is only changed, which is going to happen in this intermediary.

You have to make some changes to the tunneling thing. That is what has been done in the Tox chat. So it is incredibly reliable in message delivery for these reasons, which causes the message to reach.

(Refer Slide Time: 30:29)



Now, I even need to put a unique sequence number to each forwarded message. So, why is this required is? The destination needs to use it to reference when sending back the confirmation receipt to the intermediary. And whenever anyone gets a confirmation, it passes to intermediary two. So both the temporary copies will be removed from both the nodes and the system will keep working beautifully, providing resilience. There is a slightly different design now, but it was there because we do not have any user ID based on node IDs.

We close this lecture, and we will look into something interesting in the next lecture, which will be about how the blockchain works, which is also another peer to peer system. There is nothing as such for peer-to-peer there, but it is still an exciting topic, so I will explain that because it is also based on, it is built on top of peer to peer system.