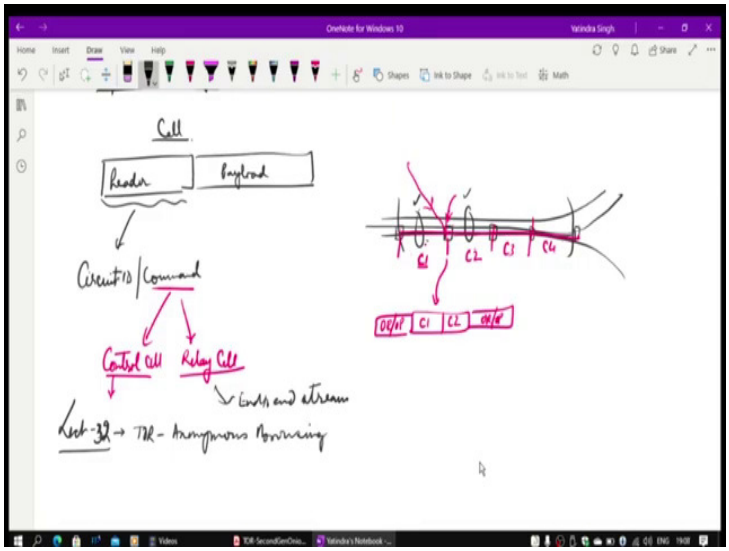
Peer To Peer Networks
Professor Y. N Singh
Department of Electrical Engineering
Indian Institute of Technology, Kanpur
Lecture 32
An Introduction to TOR Browser: The Anonymity Preserving Access of the Web Sites

(Refer Slide Time: 00:14)

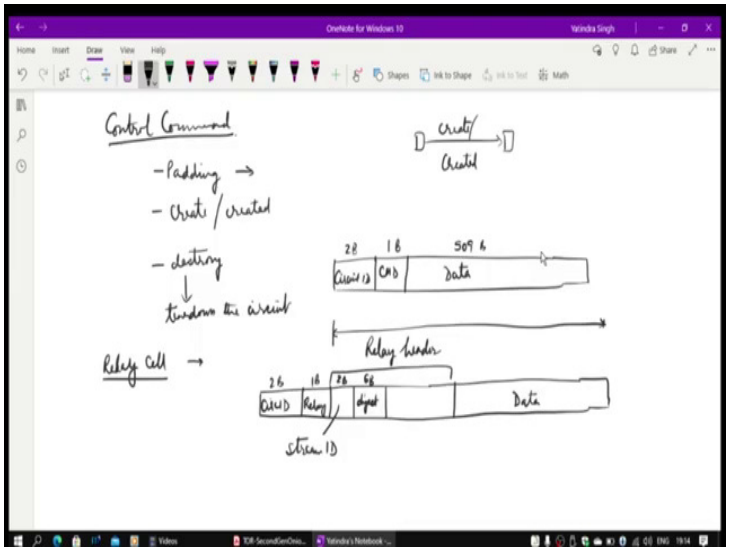


Welcome to the lecture number 32. So, I will start from wherever I left from the previous lecture. So, in the previous video, we when we closed I talked about there is going to be two kind of cells depending on what kind of command is actually being put on the circuit; one is a control cell which will be on that particular hub the whichever mode which is receiving the control sample immediately interpreted actually.

And the relay cells are nothing but something which has to go through all the way to the endpoint of the circuit which has been created. So, when it can no further be forwarded this is what is going to interpret what is there inside the relay cell. In fact, this is only now you can interpret because others cannot because it is in the encrypted form to the trees to the last Onion Router in the circuit, the circuit which has been created by the user onion proxy all the way to the exit router.

So, this basically end to end the stream data and this one is basically for the node which is the receiving it is control information is for that. The lecture-30essentially is for the TOR and moving over to anonymous browsing. Let us start on this, so we have to now move ahead with the same structure to understand this phenomenon, how the circuits have been set up extended, dismantled and how actually exactly the browsing happens.

(Refer Slide Time: 01:58)



The control commands which are available for the control cells, so let us discuss them one by one, but they are actually one command which is padding. This is normally used for just ensuring that sending a message keep alive, that you are alive just informing this thing to other endpoint. When there is no communication happening at that time this can be done and this can be used for doing padding. Basically, you are actually padding extra tests for communicating so that somebody who is trying to do traffic analysis will get the wrong information.

So that is not going to be commensurate to what actually you are communicating. So, you can put in dummy cells inside it is basically used for that purpose on a link between two onion routers or onion router or a proxy is for that purpose, second is a pair. When a forward request is basically to create a circuit is the new circuit creation it is basically request for that and when that confirmation comes from the other side other end. If this guy will send one to set up a circuit to this onion router, it will be sending, going to send a create and this will send a created.

So, as a consequence when this is done, they will both will be able to now (())(03:16) in this process a symmetric key, which is only known to these two people and this can be then used for encrypted communication between these two modes. There is a create and created, which is a confirmation message which comes in the back and then there is a destroy. So, you can destroy a circuit and totally dismantle it, is basically tear down of the circuit for that purpose.

The circuit ID and then destroy command if it is there, that circuit id will be destroyed by the other endpoint basically data structure will be cleaned off. After that if we use that circuit ID

the other guy will not knowing what to do? It will simply discard it. It is a tear down the circuit is for that purpose. Now relay cells which is other kind will consist of many possible options.

Normally relay cell will actually have some another additional header and this additional header will be there in the front of the payload. When the weight is going to happen is, I should actually draw the cell structure now, for command cell it will look something like this. There is a cell; the first part was circuit ID. Now, this is going to consume two bytes. That is a power version-2 standard actually tells is basically specification with the store was implemented there is no standardizing body, is just was implemented by volunteers. It is their respect.

You can actually decide your own thing if you are implementing your own stuff. Command always requires one byte. So, one Byte and fine the 509 bytes are required for whatever is the data which is required by this command is for that, total 512 bytes cell which is there, this is the control cell. For relay cell, before I move on to various relay commands will consist of again a circuit ID, which is again two bytes of this one byte is a command field, but the value here will not be command actually it will the value here will become relay.

I will change this will have a relay here. So, relays is thus a special kind of command if you these are not their relay will be there and then whatever is there it is not purely data. There is going to be now a header which is going to be for this what we call relay header at this additional header. This header was already there this additional header which is going to be there we call it a relay header.

And then there is a data which is going to be there. So, relay header we have again a first it will be stream ID. So, remember within a circuit there can be multiple TCP streams going, so exit mode maybe actually then setting up connections to different servers, TCP streams. So, they all are being multiplexed. This is stream ID basically used for this multiplexing purpose. So, we use two bytes for this stream ID then there are 6 bytes which are used for digest.

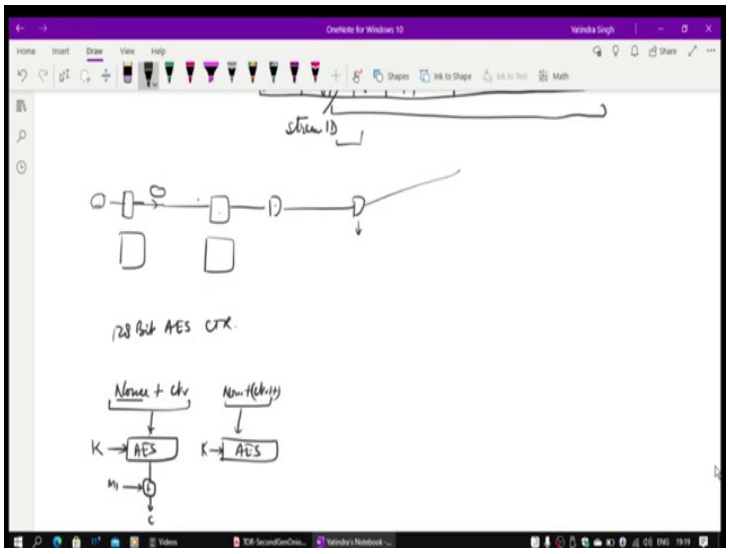
Now, this is one of the important fields. So, from this point onward to till this last point, this whole thing is always encrypted when it is being sent. This is never encrypted this part is always encrypted can be encrypted multiple times because of the onion structure. Now, this digest is essentially over this whole period, so if digest actually if you read these 6 bytes and

they match with the digest value here, ofcourse, this is not a straight forward digest they use also digest state at both sides.

So that nobody can actually to avoid certain kinds of attacks. So, these digests if it matches then this everything you can interpret, if these digest bites are not matching, then this must be still in the encrypted form and you may have to forward it. So, a node can identify whether this has already all encryption layers have been removed or not. So, when it is the first layer, it will figure out it will check for the digest not, so it has to look into the table this circuit ID on this side must have been mapped to something.

It will forward on that as it is after decryption it comes here it will also check for the digest, if digest actually matches then ofcourse, the circuit is not ending here it has to do something with this tree it has to interpret what is there in the relay field, relay header and based on that take the action whether data has to be relayed or it has to be extended. If it is not again digest is not matching, it has to look into the forwarding table, find out the circuit ID here as to mapped to which circuit ID to which particular node. Based on that, this whole thing will be further forwarded. The only circuit ID field we keep on changing as you as moving the cell ahead.

(Refer Slide Time: 08:29)



So, this is how actually we figure out the last node always the exit node, we will always will find out the digest is matching and henceforth it will figure out what has to be done. So, it may be simply relay the data onto the TCP circuit, it will simply push on the data to the TCP circuit.

Reverse side data will come, again the same procedure it will say the data is coming on this circuit it has to be converted to the transfer to this circuit, but I have to do a encryption. It will use the same encryption which has been set up between source and this node. I will talk how that will be done further. So, it comes here it will again do another encryption depending on what is the key between this and the source.

And another encryption on this side between whatever is a key between this end source and once it comes to onion proxy it will actually stripped down one after another all the layers, till it finds a digest matching is there and there is no more encryption is required. This is a cyclically, it will just keep on doing it and it has it knows that this circuit has been set up with these many relays.

This has to be known that how many times these can be done and digest should match after that. And then whatever is the data that simply will be transferred over a TCP circuit to the application that is how typically it is done. Now, this circuit will have a stream ID digest and then there is a length field which gives the length of data. So you may not all the time transporting some time it will be less also.

The maximum value will be 498 or less than value will be represented by this length. So, these are two bytes which is required for 498 because with 8 bytes you can only go till to 255. So, two bytes are represented here. Then there is a one byte which goes for command. So, this is a relay command.

There, there are various commands let us list them down what are those. So, you will have a just before moving over to commands as I mentioned this will be encrypted as I mentioned this whole thing will be encrypted actually, as it keeps on moving. So, this is only being identified relay header will only be known at the end node or only at the source node when it comes in the reverse direction.

The important thing is that how this encryption will be happening. So, for this they use AES counter based 128-bit AES encryption is a key thing, and then it is going to be using a counter-mode. What does it mean is that this whole stream whatever is being transmitted so both source and destination with the source and onion router, onion proxy and onion router within which the encryption has to happen will actually have something called a nonce, they will actually generate in the beginning of the session.

Then they will use a counter this counter will be incremented and this whole byte will now be encrypted with AES encryption, so AES encryption also takes in the key. The key is a key which has been agreed between them they say ephemeral key generated through onion keys, using TLS mechanism Transport Layer Security and this is what is the being XORed with the message, the first 128 bytes of this particular thing will be put here and I will get the cipher text.

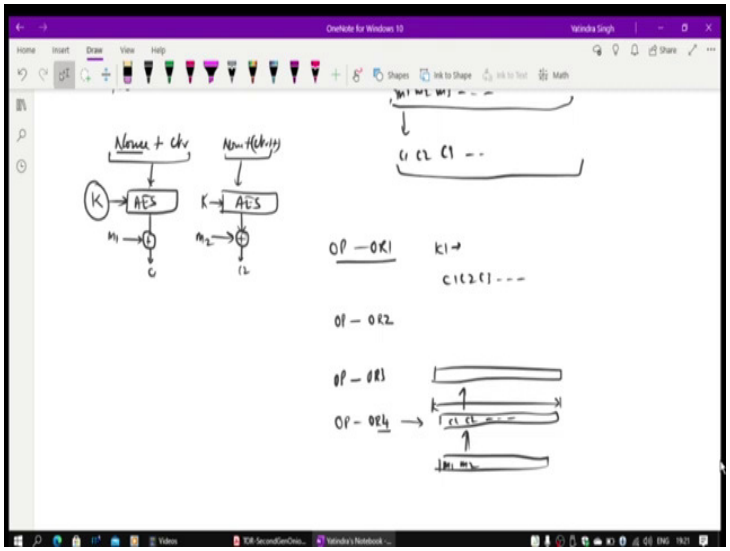
Then I will take the next one is a nonce plus the counter value will be incremented now, by one. It will be again going through same thing, now, this nonce will keep on changing every time before the stream starts once it is there, this just for that stream it will be just used both sides will keep track of their nonce. This counter will keep on going, and then after it will repeat, and most likely before that the stream will get closed.

And then of course, when the new stream is set up, this nonce have to be negotiated by both is a random number basically, which is done while key is maintained as same. This one will be coming and then I will do the XORed with the message part 2, the next 128 bits will be taken now, and so on. You keep on doing it 128 bit at a time, 128 bit at a time and so on. So that is how you kind of create a stream cipher out of the block cipher actually, there is going to be new cipher-text, what we transmit is c_1, c_2, and so on.

So, your message sequence m_1, m_2, m_3, and so on when get converted to c_1, c_2, c_3, and so on. Number of bytes which are there will remain the same. So, this is done with the key. So, between one first onion proxy and OR-1 we will be using key 1 actually which will be used for doing this encryption. So, whatever is c_1, c_2, c_3 which you have got, now onion proxy-2, onion router-2 in fact, there is the last one onion proxy-2; onion router-4 for example, in our case this is the exit router.

So, first of all, this encryption will be done the message will be converted to c_1 to and so on and this will be encrypted the message will be encrypted to this. So, this will be now with onion router-3. So, this is the forward direction going from onion proxy to exit router this will be converted to another kind of such thing, now the number of bits here remains the same they do not change.

(Refer Slide Time: 14:20)



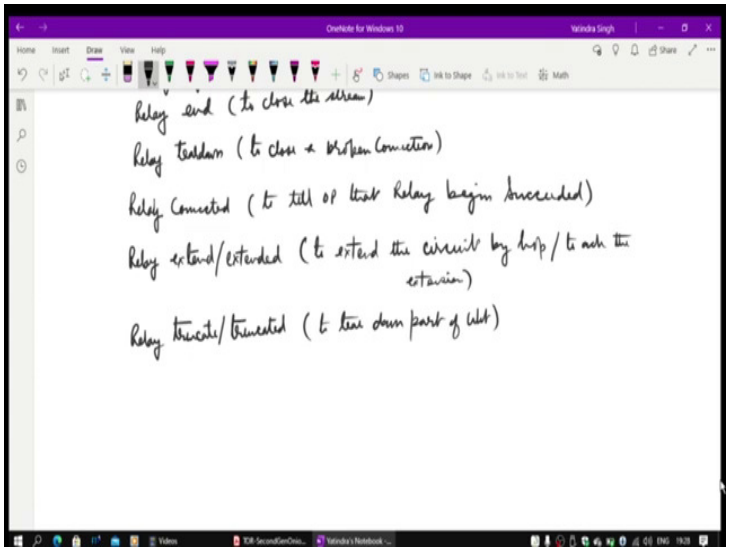
So, 498 bits will always be encrypted and this will be the for the with the OR-1. So, it will be always in reverse order. So, as they go on, so decryption will be happening, decryption also is very simple, this nonce has to be known to the other end which is decrypting it will just create this particular string by doing this encryption simply it will now do, c_1 will be XORed with whatever is the generate bit string which we are generating here the same will be fed here and then I will get message back.

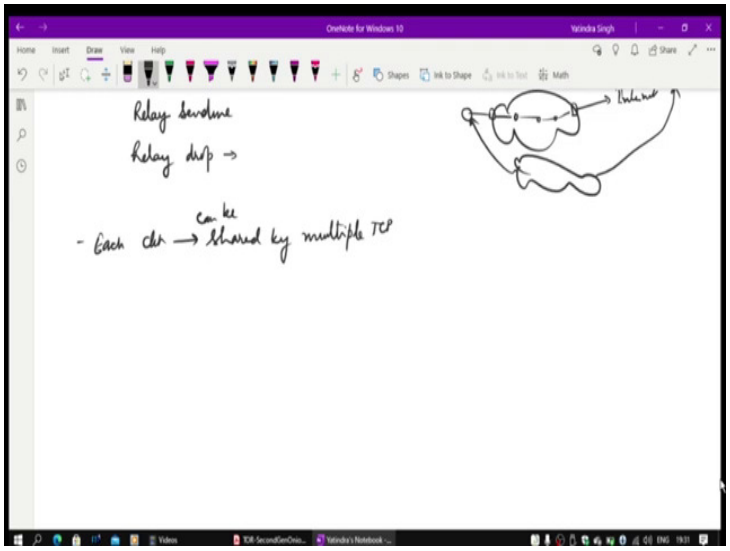
So, this is a reverse-path which will be done. So, this basically what it means that 128 bit based AES counter mode operation for encryption and decryption which is being done, they do not use cipher block chaining. Important thing is that I need not; there is no dependency. I can just simply start and compute this particular block can be decrypted independently of this. Only thing this is the random thing and the counter value. I should know which segment I am decrypting.

So non-linearly I can go to any place and decrypt it without decrypting the earlier portion that is a big advantage and is specifically used for that purpose. Now coming to the relay commands which are available, the commands which are going to be put here as one byte in the relay header. So, we will have first basic command is going to be relay data. So, this technically means you simply pass it on.

Now, remember, if this relay data will be only interpreted by the exit onion router, nobody else will be knowing it, it is a relay data field. So, because so far you do not match and you do not get your hash value the digest is not matching, you simply have to keep on forwarding.

(Refer Slide Time: 16:07)





This relay data will now push this data back into a TCP stream set up by the exit router to the server. It will just simply pass it on to that, that is a relay data, doing relaying is for that purpose. So, you can call it is for the relay flowing down the stream, relaying the data going down the stream this will happen on both sides actually. So, from exit onion router to the onion proxy also similar thing will be there, and also the forward direction also it will be there, then we have really begin.

So, you have already created somehow a circuit all the way to the exit router. Now you have to set up a new stream connecting to some server on internet, the TCP connection has to be done. So, there is a relay begins command which is used for this is basically to open a stream,

this directly remembers is the communication the relay commands are being directly between onion proxy and the exit router.

There is only communication between that, intermediate guys will not know it, then if you want this to open the stream, so there has to be something to close the stream. So, we will have a relay-end to close the stream and then we will have relay-tear down. This is basically to close a broken connection. It has to be essentially communicated to somebody whichever node and that node then can close a broken connection in that is basically used for that purpose we will see how this also works.

Then there is a relay connected, so when you do relay begin the response will come as really connected actually, this is to notify to tell the onion proxy that relay begin has succeeded. Then we have some more we have relay extended. So, you already have created the 2 or 3 hops you have created you want to create a 4^{th} hob, this basically you will not do the extension. You will just talk to that particular till that node and then that node will be communicating directly to him.

And then you can extend your telescopic encryption from there and extend the circuit switch for that purpose. You make a request for extension and the response will be if there is a confirmation it is to relay extended is basically to extend the circuit by one hop. So, one hop at a time extension is happens and this one is to acknowledge the extension, extension is happening.

Then we have relay truncated, relay truncate something which is there you want to actually truncate a circuit to for some intermediate node you can actually send a relay message you can relate directly to that guy. Remember relay message only to the intended onion router in between routers will not be known, the relay command is between always between two neighbouring routers, the relay truncate will be sent to it, so it can truncate whatever is the circuit which has already been extended from that point onwards it can be used for that purpose.

So this is to tear down the part of circuit then we also have to implement something called flow control. So, for that relay send me; this is normally sent by the exit router to the onion proxy to tell that okay do not send me now because I am already pull, I am because I am not relaying it to some TCP circuit piping it. So, I do not have that much of buffer or other guy is not consuming, you slow it down it is basically used for that purpose; how much to send.

And on the reverse side onion proxy can always tell the router which in turn then can communicate it over a TCP connection to the server actually from there or any other endpoint on the internet. Remember in our scenario, there is an exit router and then we are on the normal internet, communicating to some endpoint and this is the onion circuit, and then there is onion proxy and then this is your normal application.

Normal application had an option to go through internet directly and then communicate to server but then it is not anonymity is not maintained. So, for anonymity, I am using this TOR network, so relay sends me is for that purpose. Similarly, there is a relay drop; so relay drop is to essentially basically for long-range dummies. So, you send all the way to some node whichever is the target node till that points you will send the packets and then they will be dropped.

Basically, it is essential to create a mismatch between the traffic so you cannot do traffic profiling and cannot generate information out of that, it basically, for them. Now, there are certain important things when you create any TOR circuit to sync one circuit which has been created can be shared by many TCP streams.

Each circuit is an important thing, each circuit is shared by many can be shared. I should right can be shared need not be, can be shared by multiple TCP streams which are originating from the same source. You are actually, for example, communicating to multiple internet points. You actually use this circuit to go to the proxy and that from there it is going to that particular place. Everybody in the internet will actually see you as that persons IP address and port number, your actual port number is not available.

So, that multiplexing happens because of the stream ID which in which is there. Of course circuits are not being, you do not wait for circuits to be created, and circuits are created pre-emptively, that is very important. What does it mean actually that if you are there, you do not wait for something, you just create some circuit after identifying this is your exit router? Circuits exist, at the time some new TCP request comes in, you will now set it through this tunnel all the way to this onion exit router from there to internet, and to a server.

Meanwhile, you actually after one minute, roughly you will create another circuit, you will forget this one, every one minute, you have to now create this basically rotating the proxies we call it. You get a new circuit. Now, whatever new requests will be coming will now be exiting from here and then going to internet and then to the server. When all the TCP circuits

are closed on this circuit, this whole thing will be dismantled, this circuit will be dismantled, and you keep on generating new circuits without waiting for this thing.

And any new connections will always go on there. User would not feel the pinch, when they are communicating, they do not have to wait for the new circuit to be created first, and then the routing will be happening. Whichever is the currently available one it will just simply use that. And you keep on dismantling older ones when they expire and keep on creating new ones and all new sub TCP circuits are always to the new TOR circuits which have been created.

Circuits are being created pre-emptively, that is what it means. And user's onion proxies' job is to build new circuits periodically. So, one every one minute actually and we they are all being using multiplexing. So, multiple TCP circuit TCP connections can pass through one circuit does not matter. They will be actually carrying multiple of them. But this periodicity is one per minute actually.

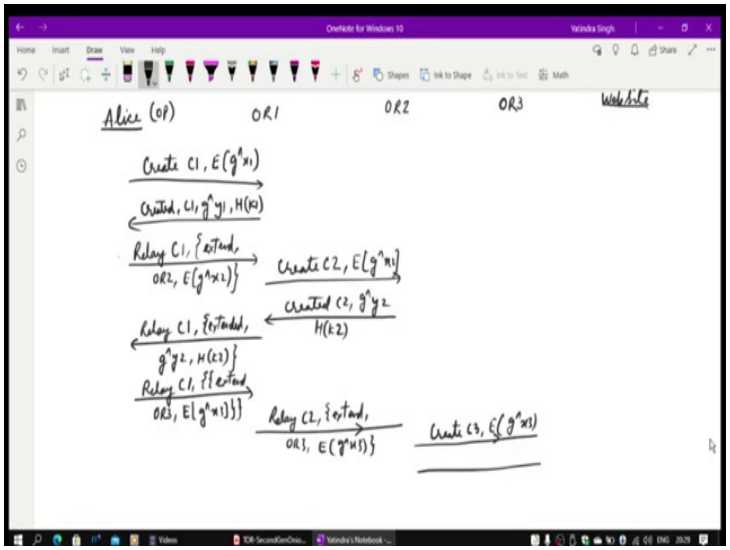
And of course, the earlier one will always be dismantled only when all TCP circuits from here are closed. If anyone of them we will maintain that TCP circuits so onion proxy has to take care of this particular scenario. Now, the advantage is this performance. Advantage is there for the user. He would not feel the pinch of this circuit, TOR circuit rotation will not be affecting the performance of the application which the user is running.

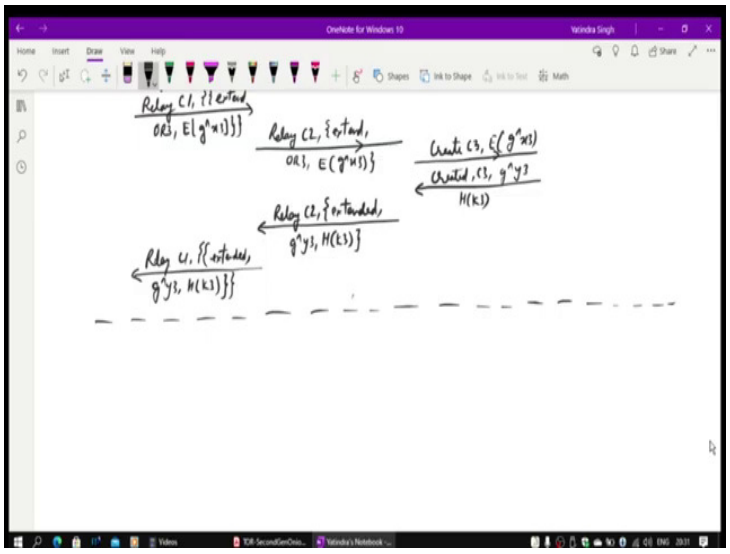
So they are all being built up and tear torn down in the background. Now, let us see how this actually process happens. So, let us look at that Alice, there Alice is the user, so onion proxy is running with the Alice, I can call it onion proxy, so with the Alice then it is I am only taking here 2 onion routers, I can take it even three does not matter. The paper which I am referring to is about the TOR version 2 basically from the Tor Browser dot projector dot org from there you can download the document also as well as the application both.

So, let us so there the example gives actually two onion routers, I can do it with three. So, let me do it with 3, from the paper anyway, you can get it for 4 also, now you can get it for 2 already 3 done but let me do something different. And there is a website on the internet. So, onion router-3 is the exit router. So, Alice somehow, first of all, connects to the directory and from there we get the list it will pick up description of the routers and will decide which websites I would like to communicate based on that will choose the routers.

And it has to choose such that they all are able to take care of the bandwidth for that which is being expected to be used by application of the Alice.

(Refer Slide Time: 26:25)





And their exit router OR-3 is willing to go to that website to the other side. I might have other criterions maybe only choosing from certain countries the onion routers. And maybe I will look at the round-trip time delay between OR-1 to OR-2 to OR-3 kind of thing.

So, and looking at all the information in the directory, so Alice will decide on OR-1, OR-2, OR-3 what are the Alice knows their IP address and port numbers on which the communication can be made directly, the Alice will not directly talk to each one of them. Alice will only talk to OR-1 and from that OR-1 will be acting as a proxy through which it

will then go to OR-2 and then OR-2 will act as a proxy through which it will create OR-3 that is what we call extension of a telescopic extension of the TOR circuit.

The Alice is the first step, will now send an, remember there is a TOR control cell, so if the control cell will be same. So, control cell we have option called create, the Alice now knows about OR-1, OR-2, OR-3, so their IP address and port numbers and they are ofcourse, onion keys are available to Alice she will retrieve it from directory server. To set up a TOR circuit all the way to the exit router and then to communicate to website there is a process.

Alice will first of all, use a path to be created to OR-1, so that knows OR-1s onion key that will send a, not a relay cell but here a command, there is a control cell which will be sent. So, this will be going at this point and it will be a command cell. This will be having a create; create is one of the commands which are available in the control cell which we will be using them.

It will also then identify which circuit number is available here. So, it will say C1. Now, it will also use the private key, part of the onion key to send a its own secret $E(g^{x}{}_{1})$ which I can write it like this, E is power g x_1 this will be used to create key 1; K 1 which is going to be for communication between manual proxy and onion router one. So, once it reaches there only OR-1 can decrypt this g raise power x_1 because it knows the corresponding private key.

It will now create another random value which is going to be g raised power y_1. Now even if it is known to somebody else does not matter because unless they know g raised power x_1 they cannot make the key which is being used as primary key which symmetric key between OP-1 and OR-1. So, this will be sent back it will be there, the corresponding confirmation command is created and the circuit ID will be C1, so this will be sent back.

Onion proxy will and of course, there will also be one more thing, the hash value of the key which has been created by combining g raised power x_1, g raised power y_1 by OR-1, that will also be sent back. Alice will also create the same key, so it will compute the hash of K1. If the both hashes matched, so this has been done successfully that said Diffie-Hellman check been done between onion proxy and onion router.

Now the symmetric key between these two have been created. Now the onion proxy of the Alice need to create extend this circuit only one part is there. OR-1 is currently the only one it has to now create a symmetric key between Alice and OR-2, but Alice cannot directly talk to OR-2 that will actually talk through OR-1 only.

OR-1 already the circuit is created it will send that relay cell to OR-1. So, that OR-1 will the relay cell will terminate and the command can be directly sent to this node. So, it will now send from here a relay message relay cell and the circuit ID is already being used, so same circuit ID has to be used here. Now, it will now encrypt because already symmetric key has been created between OP and OR-1.

That key will be used for encryption. And so the relay header, there is a relays external header which is there which will say extend they extend basically the telescopic interruption to OR-2 now, this will be sent by OR-1 to OR 2, so OR-2 will never figure out what is IP address of OP onion proxy, every communication to OR-2 or anybody else will be always through OR-1.

This is going through an extend, it will tell to whom the extension has to be done, it will be OR-2, so Alice has a specified and it will send the, it knows the public part of public key of part of the onion key of OR-2, so that will be used to send another random number so that a symmetric key between OP and OR-2 can be created. I can call it g raised power x_2 will be sent which is in encrypted form and this is using the public key part of the OR-2.

Only OR-2 can decipher this value nobody else. Now this encrypted for OR-1, symmetrically OR-1 receives it, it will decrypt actually. Once it decrypts it figures out that digest is matching actually because this is only being encrypted once. So, once the digest matches it will now interpret what is there inside the relay header, it says it has to be extended extension request, it has to go to OR-2 and this is thing which has to set, so it will now actually create a circuit using a command cell.

Now, command cell remember is always between two more onion routers or onion proxies. Onion proxies and onion router pair which are directly communicating, you cannot relate. So, here it has been relate, and now there is a direct communication, it will send a create now there is going to be and that, it will find out which circuit ID is still not been used here.

Because, OR-1 to OR-2 route can also be used by somebody else for their own circuit. Any unused circuit number that will be used and a routing table at OR-1 will be created which will say map from C1 coming from OP to OR-2 with and it has to be converted to C2. That is only information which is available nothing else and while coming from OP, I have to do a decryption, while sending to OP to do encryption only, that is information known to OR-1 as of now, which is going to be kept in the table. Now once this creates, C2 has been sent, it will send also this information.

Now OR-2 is being used by OR-1 so that to whom to communicate C2 is identified and this thing entity is a data for the extension which is sent as it is. Once OR-2 receives it, it will be able to decrypt this one using its private key and it will now generate another random number, it will send it back, it will be created of course, confirmation. Circuit ID will remain same, it will send g raised power y_2 and of course H(K_2) will be sent back, the way it has been done here.

Once it reaches here, OR-1 now knows that C1 as this extension has been created. It will simply now send this information back after doing encryption. It has to be relayed back now. And if now C2 has to be mapped to C1 that is now in the routing table of OR-1, to creates C2, it will do the encryption with the symmetric key which has been already generated after this step between OP and OR-1, It will be now extended that is a confirmation which happens.

And the relay header and it will now send the information back and H (K_2), this 2 will be sent. So, extension has been successfully done. Alice onion proxy will now decrypt it and will know that yes, extension has been done. It knows now g raised power y_2. It already knows g raised power x_2. It can now find out the key between OP and OR-2. Now a path has been set up OR-2 is still not able to know what his IP address and port number of onion proxy.

Alice is still unknown to OR-2 but Alice knows about everybody who is there on the way. Once this is done, but now my exit route is OR-3. I have to do something more I have to do one more extension from OR-2. How that will be done? We will, another request which has to go, it will again be relate and it will be now has to be deciphered by OR-2. I will do a double encryption. Sorry this has to be C1, not C2.

Relay C1 and it will now do a double encryption. First encryption will be the key which has been now created after this step with OR-2 and then the key which is being there already being there with the OR-1. So, twice, it will be doing. It will again make a request for extend. And this request now technically is for OR 2. OR-2 has to go to extension and it will save it to whom it has to wait, it will specify OR-3.

It will also send using the public key part of the OR-3, it will now do the encryption and send its own random string. So, I can call it x_3, and remember it is a double encryption that is why two twice curly braces have been put, once it reaches here OR-1 will decrypt it. It cannot make any sense because the digest is not going to match. Once digest does not match it will

simply forward it to the next request, it will be now will identify only C1 it has to map to C2 and send it to OR-2, and it will simply do that.

And decrypted part it will simply transform because it cannot understand what is there in the relays payload. It is one encrypted form it will go extend it has to be OR-3 its encrypted with the public key part of the OR-3 actually, now OR-2 will now will be able to decrypt it and its digest is going to match, once the digest matches. It has to interpret what is there in the relay header.

The relay command is now extend. It has to do extension to whom and this key has to go as it is. So, it will now do and the command actually now here, because here the relay circuit is now terminated. Now the command has to be there for, cell has to be sent for the extension. Now, it will send again a create from here. And it will call create, and whatever is a circuit ID available on this route that will use, let us call it C3, it can be any value, I have just put it C1, C2, C3 that came to mind.

And it will now say create C3 and it will just simply transfer whatever was received or neither OR-1 nor OR-2 knows about what is this value. This will only be not deciphered by OR-3 using its own private key. Once that is done, it will now successfully create the symmetric key to be used with Alice. In fact, it does not know who the Alice is it only knows that there is a circuit ID for this I have to use this symmetric key that is it.

And C3 to C2 this know only that mapping it will just do one other encryption wrong, while sending it back. None of OR-2 and OR-3 knows about Alice as of now. So, it will send it back, and of course, is the response will be created C3, it will be sending another random string and H(K3), as usual as the way it happened earlier. At this point, now OR-2 will now be further encrypting it. And it will now do use relay, because now extension has been done.

It only knows C2, C3 has been done is successfully. It will now simply create a header, response header for the relay cell, it does not know it has to go to Alice it can only just use this C2 and send the confirmation. It will encrypt with its private its symmetric key which has been created with the Alice and write extend it which is confirmation.

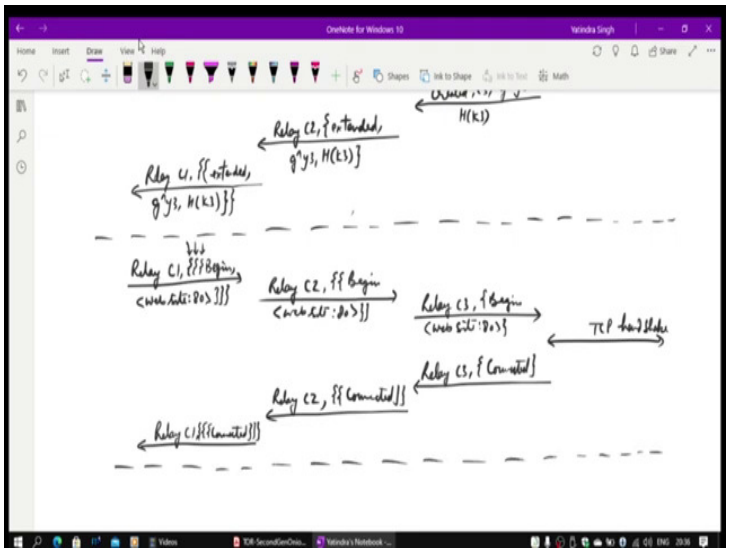
And it will now send g^{y3} which has been received as it is plus H(K3). And this encrypted thing will be received by OR-1. OR-1 will also cannot interpret anything out of it. It will simply C2 to C1 and of course encryption because that is the way it has been noted down, it

is in its routing table. It will now say simply send a relay message C-1 and double encryption, one more time. It will be done here.

And it will be extended rest everything remains inside the same. So, g raised to the power y_3 is only known to actually OR-2 and it will be only known to this thing, but g raised to the power x_3 will only be known to OR-3 and the Alice, they will only be able to get the symmetric key between OR-3 and Alice and they only can use it.

Once it comes here at the circuit has been now created, circuit has been set up. Once the circuit has been set up now, you can actually, you can use relay command it can actually relay the command all the way to the onion the exit router which is OR-3, and ask him to set up a connection to a website.

(Refer Slide Time: 40:59)



Now ,the relay command which will be used will begin to set up a TCP circuit actually. It can now send from here the relay command circuit ID will remain C1. Now, how many times it has to be encrypted once for this, once for this, and once for this, so three times. First encryption will be with the symmetric key with OR-3 then the OR-2, and then whatever is with to OR-1 in that sequence. The outer one is with OR-1, this is with OR-2. This is with OR-3. And then ofcourse you have to tell what the relay command is and what are the arguments for that command will go in data.

In this case it will be website to whom you want to connect and of course, these are the closer for encryption received by OR-1. So digest is not going to match, so it is certainly not for it. So, it has to now find out C1 has to be mapped to whom; C2 which has to go OR-2 that is as

per the routing table which is maintained. It will simply decrypt one layer and send that further. It will just do the relay again, C1 is mapped to C2, so, C 2 will be there.

Now layers will, one layer will be taken out rest everything remains as it is. This is a command and is the data which is going to be there in the data part. Once it reaches OR-2, OR-2 also cannot decipher anything because digest is not going to match, so, it has to simply look C2 has to be mapped to C3 and sent to OR-3 it will do the same. So, it will now send again relay; C3 comes from the routing table OR-2 and one encryption layer being removed and it says the relay command is this begin in the website.

So, OR-3 will decrypt it, the moment OR-3 actually sees digest is going to match. It will interpret what is a header, so, header says you set up a connection stream to a website which has been given. So, it will now set up a connection, it will negotiate a connection actually with the webserver, so it is a TCP handshake will be done.

Once TCP connection is set up then after whatever is actually coming from here will be just routed there. You will not be sending begin actually now. You will be doing relay data. So that is the command which will be coming so that relay data once it comes it simply take whatever is a payload and pass it on to TCP cell.

In the reverse direction the same thing is going to happen whatever is coming has to be connected to you create a relay C3 and data command and whatever is a payload byte, can we just simply put here and sent in the backward direction. This is the way it is going to happen. Now let us move ahead further. Once the data comes in connection has been cleared. It will send a relay.

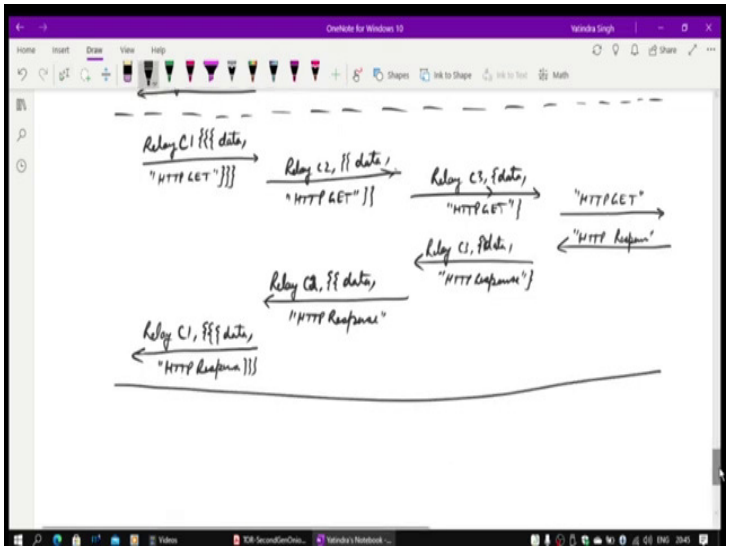
Begin has to be come from so again there is a relay command, in this case. It will be connected, and it will be closed one encryption. Once it comes here, this guy cannot figure out anything the digest does not match at least not for him. It will simply do a further encryption by looking at the routing table and circuit it will be changed to C2 and it will become connected. It comes to OR-1, OR-1 also cannot figure out, look at the routing table and send it further. It comes to the Alice essentially.

What the Alice will do is, now there is going to be 3 times the encryption which has to be here. Its digest will not match. It will keep on going decrypting ignores the complete sequence. So, it will do in the reverse sequence, first of all, using symmetric with OR-1 then symmetric key with OR-2 and then symmetric with OR-3, it will decrypt three times digest is

going to match and it knows yes connection has been set up and now communication can happen.

It will just inform the browser that TCP connection has been successfully done, the socket is okay, now you can start setting up. The third phase is now data communication which will happen.

(Refer Slide Time: 45:31)



In this case here the Alice will now send, for example, get something get a page from that website. So, connection has been done with the website, so it can send a relay message but remember this relay has to be done till the exit router and then from there, it has to be simply by piped on to the webserver.

Relay C1 and now the command which will be sent it will be encrypted three times and it is now a data command. A relay command is relay data which is being used which basically means at the exit router whatever is there is simply pipe it onto the TCP connection it has already been set up. So, maybe the command which has to go is only HTTP GET and whatever other things, URI or whatever it is, the file which you want to access this is what we be generated by Alice.

This comes to OR-1 it cannot make any sense out of it digests does not match. At best it knows that C1 has to map to C2 and send to OR-2 that is the only thing known. It comes here and it will just simply swap C1 by C2 remove one encryption layer. Only two encryptions will be available inside command and whatever the payload that remains the same is. What

we will also not figure out it will just remove one encryption layer same thing and it will just simply again do the relay.

This end to end communication is being created between Alice and OR-3. This C3, there is only one relay one encryption layer now left. The OR-3 will immediately will retrieve digest is going to match once it actually removes the encryption and once digest matches. It has to look into, what is the data, what is a command, relay command it says data. Whatever data has come in the payload simply has to be sent to the server and this command says HTTP GET.

Now server actually does not know who is the guy actually who is fetching it. OR-3 also does not know, OR-2 also does not know only OR-1 knows, which this is unless all these three guys are actually being accessed by any surveillance agency. They cannot figure out who is talking to the server. In fact, these guys will also do not know which server is communicating only OR-3 knows that to which server the communication is happening because it is acting as a proxy.

Only OR-3 can figure out and as I know you can do traffic analysis here and traffic which has been generated here, someone you can match the statistics. You can kind of get a circumstantial evidence of this being related to this exit mode, but because you are changing every one minute. This becomes extremely difficult to correlate. Now the HTTP response will be sent back by the server. Once it comes, it knows that this particular circuit, TCP circuit has to be mapped to this.

It does not know who for whom actually it is as of now. It only knows that I have to send it through relay. That is the only information it will just do C3 and kept the HTTP response. It will also now make it a data actually, is a relay command is data here, data has to be simply passed on and response the payload part will contain HTTP response. It will come to OR-2 does not know anything, it will simply further encapsulate it after one more layer of encryption, this will become C2 sorry.

And one more the command and the payload part remains as it is inside the relay header and relay payload. The OR-1 will also do the same thing. It cannot interpret anything out of it. At best it can simply relate back as per the table. There will be three layers now. The HTTP response is now being received. Now, this actually Alice knows that I have used three onion routers to go to the website.

So she will actually do three times decryption in the same order. And after that the digest is going to match, it immediately knows that there is no problem, it will now go the HTTP response is a relay data, it will simply pipe this response to the browser which is connecting to the onion proxy, so your browser will start doing browsing anonymously over the internet. So, nobody on the internet knows who is browsing and you are browsing anonymously, no intermediary knows except your next guy, but what you are browsing is not known to anybody actually.

Now, this digest mechanism also provide, as I told that, it is possible that if you do traffic analysis here, and if you do traffic analysis here, what Alice is whatever is exiting from Alice if they are matching, I can actually kind of correlate them. And then figure out that what most likely you are browsing, that is a statistical analysis that also can be broken. So, this actually provides something called leaky circuits.

So, in this case, what you do is you not only actually transmit data for this you can transmit for data for some of the circuits for example, you can instead of doing twice the encryption only thrice in this data you do twice actually, in fact, when you are setting up the circuit stream here. So, when you are doing this begin, so instead of if actually start instead of going encryption thrice if I do it twice, it will exit. It will actually exit here.

And then if we are actually if I give begin connection, so, it will set up a TCP connection from somewhere here that will now remember that C2 whatever is coming this circuit, this particular stream ID has to be connected to here Now, remember this begin is mapped to the stream ID. There is one stream which can go all the way here one stream can exit here one stream can exit here.

My traffic here will be different than what you are going to measure here, it is going to be very difficult from where things are emanating and you cannot correlate it. It becomes extremely good in actually maintaining anonymity. Now, important thing is that this additional header can only be interpreted at the destination node nowhere else that is very important, and depending on how many layers you have done, you are deciding which one of the intermediate nodes is going to act as an exit node.

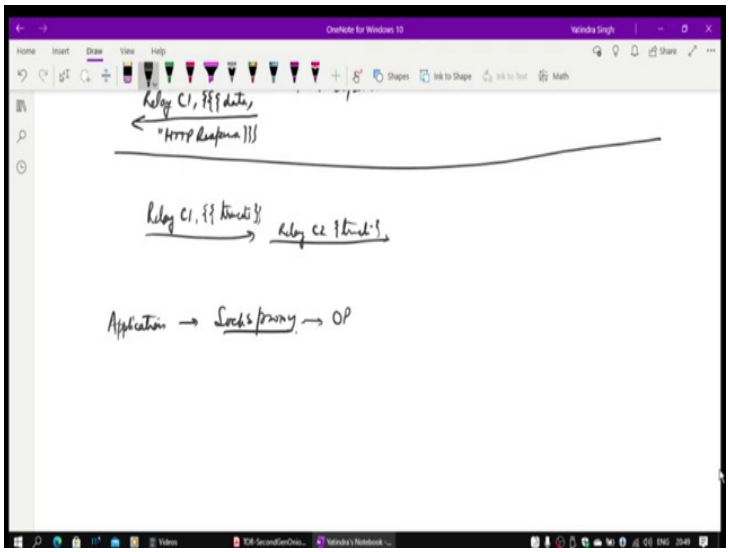
And of course, if there is somehow the digest does not match and you do not have any information of how to forward it, okay if OR -3d oes not know, there is a thing and you send a relay data which is terminating here and my digest is not in fact digest is not matching, you

do not know how to map it further. In that case, simply you have to, this circuit has to be dismantled. It will simply send failure.

It will simply send abort and then this particular circuit to be dismantled and new one will be created and it will say restore control cells will be sent back and it will be clear down. Similarly, in the reverse direction, if this thing happens that your digest does not match after doing three times decryption. It will simply tear down the whole circuit. And of course, not only this we can do incremental destruction also, this is done through truncation.

So, if for example, I want to truncate this particular part, so, this dot 3, so, in that case, the truncation has to be done Alice can send a relay truncate message.

(Refer Slide Time: 53:29)



So, it will be, it can send a relay on the same circuit ID and it only does twice the encryption. OR-2 after that the truncation has to happen and it can send a truncate. And the truncate message has been done. This will go here, from here and then again this will be mapped onto because, after one encryption is removed, it cannot make sense out of it will simply map to C2.

And it will go with truncate it will be reach here and now you can interpret this a truncate request for C2 it will remove from that routing table entry and it will also send the command for in that case for tearing down the connection. So, remember there was when we were looking at the control cell there was third command which was destroyed. It will send a destroy thing and destroy command will keep on send.

All the circuits will be cleared after that and once this is truncated, this is the only point. Now Alice needs to remember this has been truncated and this circuit ends here. It can do a further extension in some different direction. It can dynamically keep on destroying or truncating and keep on extending in a different direction. So that possibility does exist that gives a lot of flexibility. And of course, there is one more thing if, for example, this node OR-3 goes down then what is going to happen?

In that case the previous node will figure out this node has gone down. And it can actually it will send a truncated response can be sent back to the Alice and this routing table entry can be purged and now. Alice will know the truncation has happened from this node. So, my circuit is only up till this point and it can start an extension in a different direction or will only use this as an exit route.

Now, the one of the important things there is one attack, for example, I can just simply take down one onion router node. This will break down the circuit now, that is taken care of because the earlier guy will send truncated and I can actually now extend in different directions then start again accessing the internet anonymously. That you break a node and circuit goes down that kind of attack can also be taken care of.

Normally, idea here is that this was done for the browser, it can be done for any application actually, you can do TELNET. So, you need the TOR project dot org gives only the browser which ones over the TOR, but you can actually integrate TOR library with any application. Normally, any application, you do not have to do much modification, you can use SOCKS proxy, the application can connect to SOCKS proxy. And using this SOCKS proxy, you can connect to onion proxy actually.

And this will then create all the onion circuits for you and then you can connect to the server on the other side, and the key circuits can be used to ensure that you can break down that traffic correlation attack. And this is how basically this whole system works and it can work for almost anything. One of the common which remains is the speed. How to handle this speed; because this mostly network is very slow.

I am not going to go into details of this. So, next video I will be talking about hidden services or what we call the Darknet websites how those are actually gets created. And nobody knows that we have these Darknet websites are actually being hosted. So, we will look into that how that thing is done. That is also very interesting design, which has been implemented in the TOR system.