Blockchains Architecture, Design and Use Cases
Prof. Sandip Chakraborty
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 50
Research Aspects – III (Collective Signing)

Welcome back to the course on Blockchain. So, in the last class we have looked into the Bitcoin NG mechanism in detail. So, which tries to solve the scalability and a fairness issues associated with the standard term Bitcoin proof of work base system. Now Bitcoin NG also has certain problems, because it is ultimately compromising with the security of the system. In the sense that whenever a particular minor is making a leader getting the leadership of a key block, we are giving the power to that particular minor to introduce multiple microblocks in the system.

Now, whenever we are giving that power to that particular minor to introduce multiple microblocks in the system; obviously, that is a level of compromisation which is happening in the system. In the sense, that well now you can just think of a scenario the solution I will come up later. So, the scenario is something like this; say, assume that I am a compromised minor, and what I am trying to do that well, I got to hold of a key block, that is good for me, now I will be able to generate multiple such microblocks one after another.

Now, for the first few microblocks say, within the time I had made a estimation that well I will be able to generate some 12 microblocks. So, what I did that well for the first 8 microblocks, I have made them correctly as a normal microblock or as a genuine or a valid microblock. But the 11th microblock that I am generating I have making some tweaking inside that. So, I am say possibly introducing a double spending inside that microblock, or I am inserting some invalid transaction inside that microblock; which does not get validated with the existing serialisation strategy of that transactions.

Now, if it happens well, the system will be able to detect that no doubt, whenever the verification will be done by other nodes to add up that microblock. But the problem here is that by the time you are detecting that the serialisation order of this microblock is not valid, you have already added multiple other microblock from that compromised user.

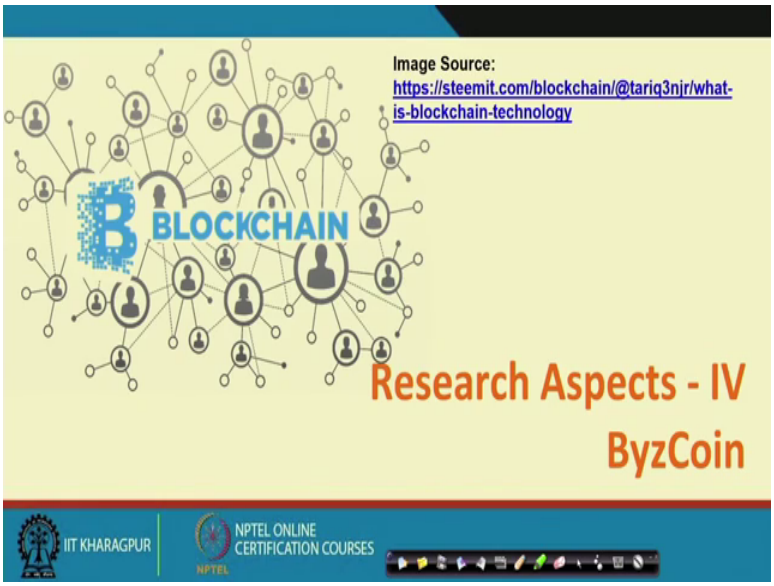
So, that indeed compromising the system the entire system which can somehow make a make certain kind of attacks which we call as a fork attack in the system.

So, you can generate so now, all the microblocks which has been generated in the system, they will be become a fork because they are not remaining valid. So, the last microblock not remaining valid; that means, you will be add up another key block by another minor; which will take care of the charge of leadership. But the additional microblocks which has been generated by that time, they will become a fork to the system.

Now, here the problem is that well certain malicious user they can launch a Sybil attack on the system. So, if you remember the term Sybil attack; where malicious note, it generates multiple identities, by using those multiple identities they try to participate in the system. Now with the help of that Sybil attack, it may happen that well a malicious user continuously generating this kind of forked microblocks, which are not part of the original system, but as the end user as a client or for the other miners; ultimately, they have to process those microblocks, they have to check the validity of those microblocks.

So, the problem here is that, you are not checking the validity of the system at the beginning, but you are checking the validity of the system at the end, after the system has already added up the nodes or the microblocks in the blocks. Now to solve this kind of compromisation and also to look into the scalability aspect of the standard proof of work based protocol, people have explored further and multiple other protocols came into existence. So, today we will discuss another protocol and before going to that we will cover a certain preliminaries which will be required to get basic knowledge of that particular protocol.

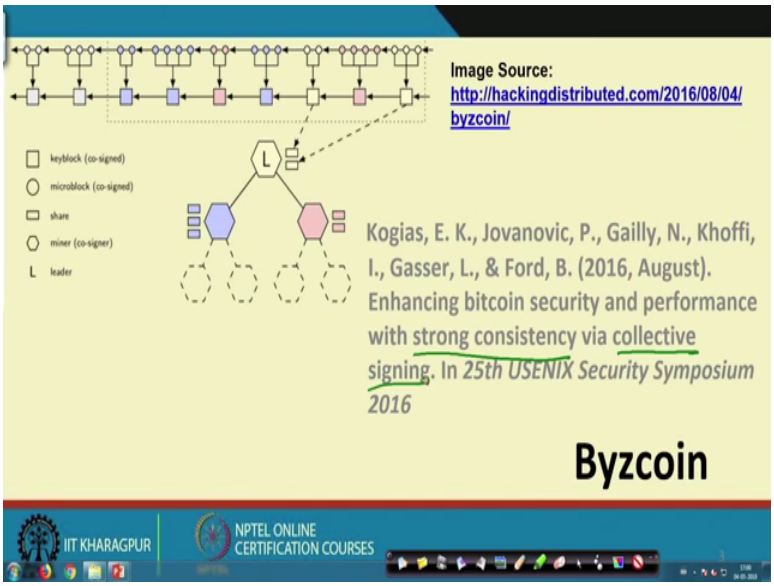
(Refer Slide Time: 05:05)



So, the idea of that protocol it is called as Byzcoin, or this protocol is indeed or nice combination of proof of work and PBFT based systems. So, whenever we look into the details of Byzcoin you will find out that well we are ultimately introducing the concept of PBFT inside proof of work and we are deriving a consensus protocol which is a combination of proof of work and PBFT. So, let us look into this particular protocol in details.

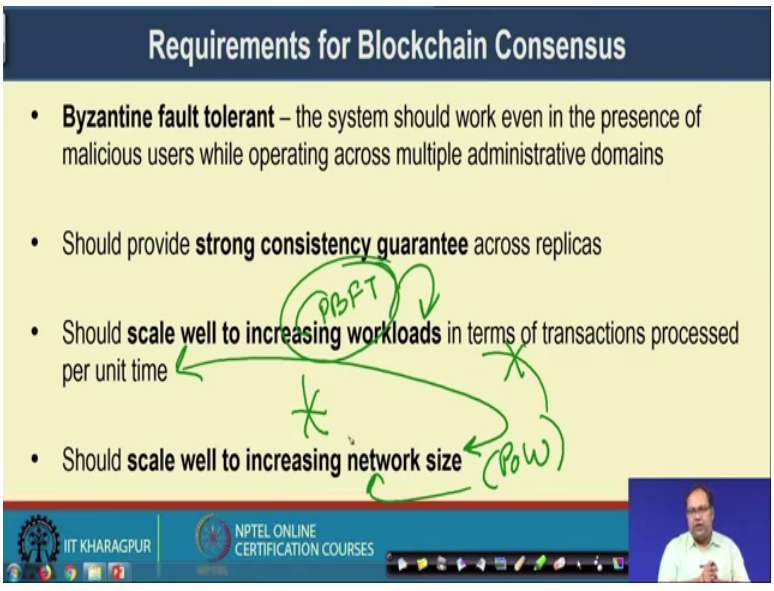
So, this Byzcoin paper that was published by Kogias Jovanovic Gailly, Khoffi, Gasser and Ford in 2016 that was published in 25th USENIX Security Symposium and the title of the paper was Enhancing bitcoin security and performance with strong consistency and collective signing.

(Refer Slide Time: 05:53)



So, there are 2 keywords in this particular paper. One is the strong consistency, and the second keyword is this term collective signing. So, let us look into these keywords first, that what is mean by collective signing and strong consistency. Then we look into the protocol in details.

(Refer Slide Time: 06:15)



So, if I go reback to the requirement of a blockchain consensus protocol, we have primarily 4 requirements. So, the first requirement is that the system need to be byzantine fault tolerant; that means, the system should work even in the presence of

malicious users, while operating across multiple administrative domains. The second requirement is that it should provide strong consistency guarantee across the replica. So, the by term strong consistency guarantee across replica we say that well whenever you are creating multiple copies of the blockchain and every individual minor or user whatever you say them, they are maintaining multiple such copies of blockchain.

So, we require that well there should be a strong consistency support across those local copies of the blockchain. So, everyone should possess the similar copy of the blockchain and that is the validated copy. So, that way we provide the kind of strong consistency support across the entire system. And on other hand, the strong consistency support also helps you to prevent the fork in the system.

So, whenever you have multiple fork in the system that is the kind of weak consistency. Weak consistency in the sense, it may happen that well a note can hear 2 different copies of blockchain. And whenever you are hearing 2 different copies of blockchain, ultimately you are accepting the one based on this 50 percent or better to say 51 percent majority voting. So, the copy whichever you are getting from more than 50 percent of your neighbours that you are accepting as a valid copy, and then you are propagating that copy further.

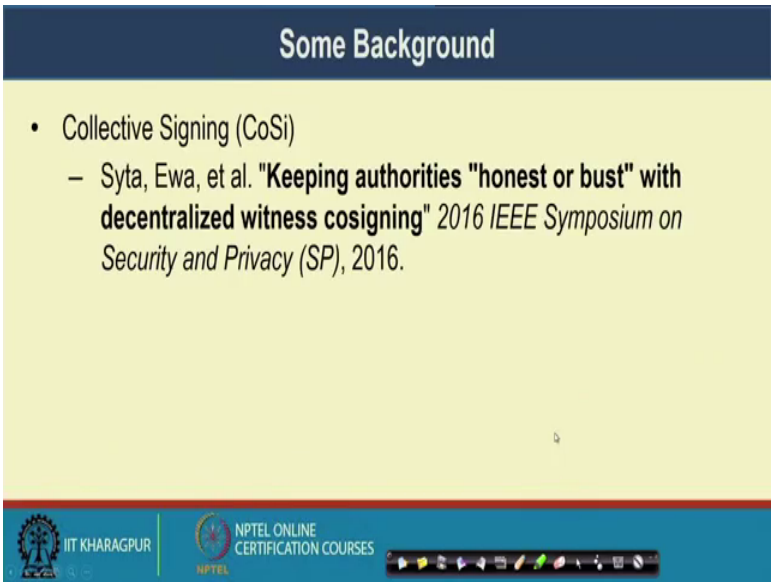
So, that way the valid copies are propagated in the network, but there is always a possibility that well 2 different copies of blockchain are there at 2 different end of the globe. And that way proof of work does not support strong consistency; rather proof of work support a weak consistency. So, eventually you are ensuring that everyone will work on the longest chain, but if you take any short term in instances within that short term instances, it may have happen that 2 different copies of the blockchain are being used at 2 different part of the globe which are kind of far apart.

Now, with this particular case our requirement is to always go for strong consistency. So, this PBFT type of protocols they always support strong consistency in the system. The third requirement is the system should scale well with increasing workload in terms of transaction processed per unit time; that means, it should have good performance in terms of transaction throughput. And the final requirement is that it should scale well with increasing network size.

So, if you look into the proof of work based system so, and the PBFT based system there is a clear trade-off between this requirement 3 and requirement 4. Indeed, this requirement 3 and requirement 4 is kind of conflicting based on the current set of consensus protocols that we have. So, if you look into the PBFT type of protocols, the PBFT type of protocols falls in this group. Whereas, the proof of work type of protocol falls in this group. So, the proof of work type protocol, it scales well with the increasing network size, but it does not scale well with increasing workload. Whereas, PBFT type of protocol they scale well with increasing workload, but they do not scale well with increasing network size. So, we have a kind of trade-off between 2.

So, these are the 4 basic requirements of a blockchain consensus protocol, and the Byzcoin protocol actually tries to solve, this tries to provide these a byzantine fault tolerant system, a system with strong consistency guarantee. A system by having a proper trade off or a balanced trade-off between scaling well in terms of workload; that means, the performance in terms of transactions per second that can be supported, and scalability with respect to number of increased network size. So, that was the first requirement from the requirement perspective.

(Refer Slide Time: 10:53)



Now this entire Byzcoin protocol that is based on a concept called collective signing. So, this concept collective signing that was proposed again in the year 2016 at IEEE Symposium on Security and Privacy, title of the paper was keeping authorities honest or

bust with decentralized witness cosigning. So, idea of this collective signing is something like this, that well among a group of peoples, certain peoples can behave maliciously.

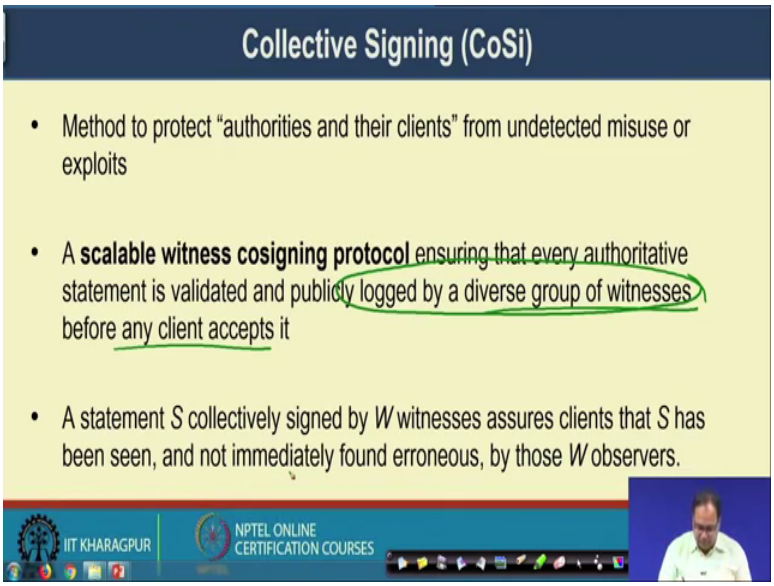
Now, if you are getting signatures from all those people in the group, and if you are having a scenario; where you believe that well, more than 50 percent of the persons in that group, they are valid persons, or they are not malicious users; that means, less than 50 percents are malicious users. Then ultimately you will get a document with signatures from most of the valid nodes or most of the non-malicious nodes and non-malicious authorities.

So, that was the concept of collective signing, that rather than signing the document from a single user, you rather sign it from multiple user; so, that at that particular place you can prove the validity of the system. Now if you think about why collective signing will be important, or how collective signing will change the shortcomings of Bitcoin NG. So, in case of Bitcoin NG you have a single signer; that means, the minor who is get who is generating the key block. So, that is the single signer who is signing all the microblocks, and that is why there is a possibility that if that signer is compromised, then there is a possibility of having compromised microblocks or invalid microblocks coming in the system.

But rather than signing that particular microblocks by a single signer if you make them sign by a authority; that means, a set of signers rather than a single signer; obviously, among that set of signers one can work as a leader, but if you are making the system to get signed by multiple signers rather than a single signer, then at the beginning itself you will be able to prove the validity of a microblock. So now, you know that that microblock is not compromised, because it has been not signed by a single leader rather or single signer, rather it has been signed by a group of signer.

And in that group of signer less than 50 percents are malicious. There cannot be a scenario when more than 50 percent of the signers are malicious in the system. So, that is the broad idea where this collective signing concept can solve the problem in Bitcoin NG protocol. So, let us look into the details and go deeper inside this collective signing protocol, the collective signing mechanism.

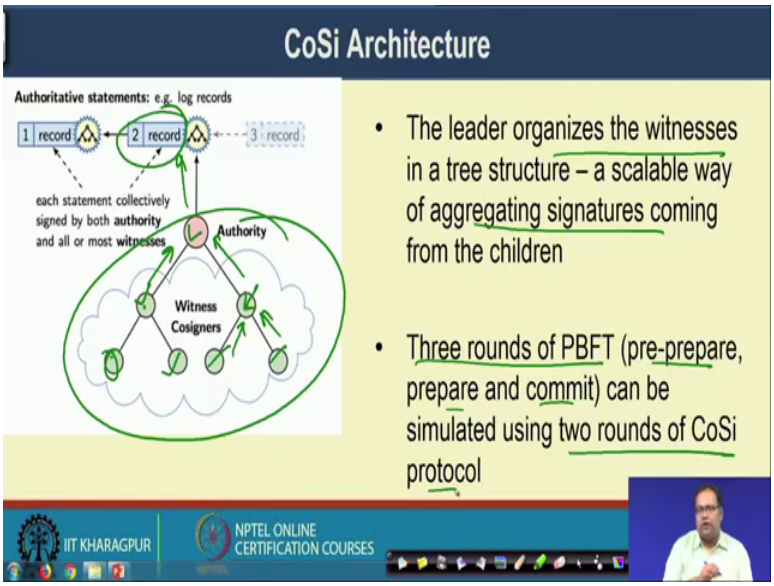
(Refer Slide Time: 14:07)



So, in summary this collective signing, it is the method to protect authorities and their clients from undetected misuse or exploits. So, you are not making one signing authority, rather than having a single signing authority. You are having multiple signing authority who are signing or who are providing the validity of a particular aspect or a particular document.

So, it is a scalable witness cosigning protocol which is ensuring that every authoritative statement is validated and publicly logged by a diverse group of witness before any client accepts it. So, you are ensuring that this particular statement is being validated by a diverse group of witness. So, mathematically a statement is collectively signed by w witnesses assures client that s has been seen and not immediately found erroneous by those w observers ok.

(Refer Slide Time: 15:17)



So, this is the architecture of the CoSi protocol. So, whenever you are adding up certain records here, rather than this record gets validated by a single signer which is there in case of Bitcoin NG; where the minor who is generating the key block it is signing all the microblocks, you are signing it by a set of witness cosigning. So, they are working as the authority for signing this particular record. So, that way this record has been validated. Now this record you can here in terms of blockchain. This record is a set of transactions. So, this set of transactions has been validated by this entire authority.

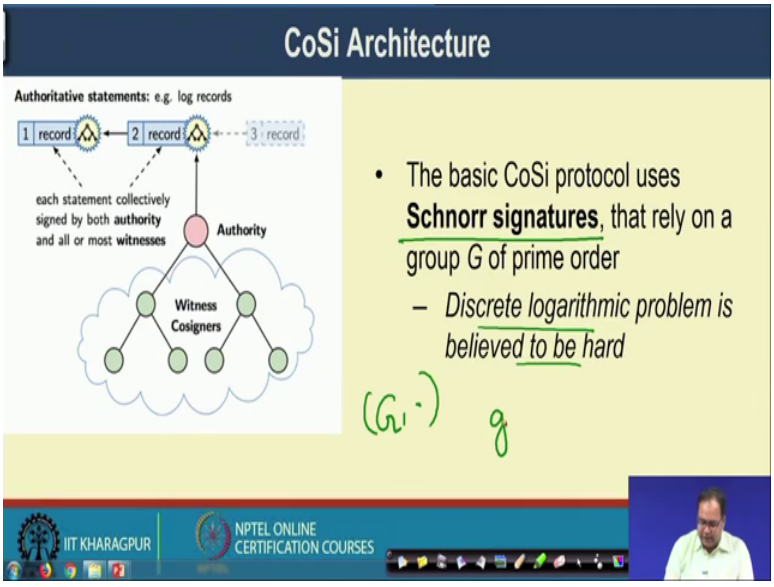
Now, this leader so, you have a leader node here. So, this leader organizes the witness in a tree structure. So, we organize the leader in a tree structure to have a scalable way of aggregating the signatures, coming from the children. So, it is just like that this particular record will be validated by this node, then by this node then the sign from this node and this node will get aggregated at this node, and it will add up it is own signature.

Similarly, this node and this node will add up their signature independently, they will come here this node will put it is own signature aggregate the signature which is coming from this 2 node, then these aggregated signature will be forwarded to the leader. Leader will finally, put his or her own signature and aggregate the signatures coming from the children and that will be work as a final aggregated signature for this record. So, rather than having the records signed by a single signing authority, we are signing it by multiple

authorities with the help of this aggregated signature. And this tree structure will actually help you to aggregating the signature in a nice scalable way.

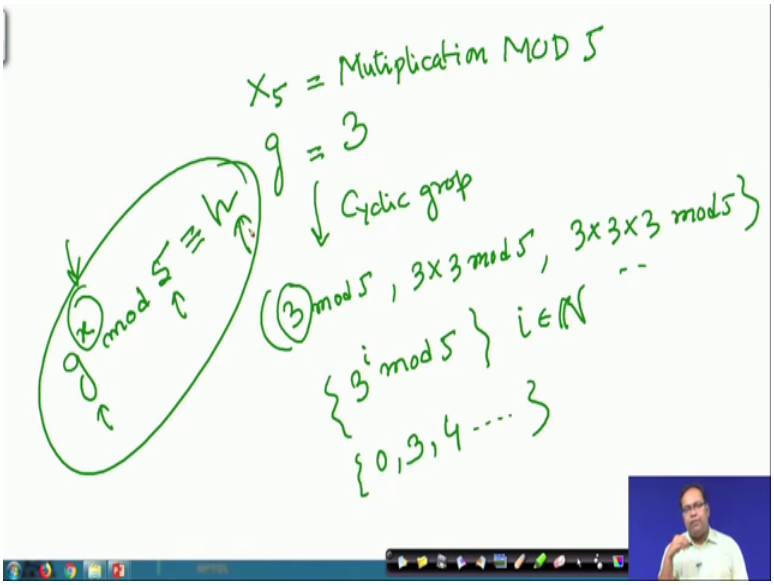
Now, interestingly I will we will see that later on that 3 rounds of PBFT protocol, like the pre prepare and the commit protocol can be simulated using 2 rounds of CoSi protocol. So, if you run 2 rounds of CoSi protocol it will be equivalent to having 3 rounds of the PBFT protocol. And that way using CoSi we will be able to simulate the PBFT behaviour.

(Refer Slide Time: 17:37)



So, this basic CoSi protocol it uses a concept called Schnorr signature or Schnorr multi signature; which rely on a group G of prime order. So, on that particular group the discrete logarithmic problem is believed to be hard. So, what is the discrete logarithmic problem on a group G. So, you have a group G with certain operation. So, we say this small g as a generator of group G. So, what is a generator of group G?

(Refer Slide Time: 18:19)



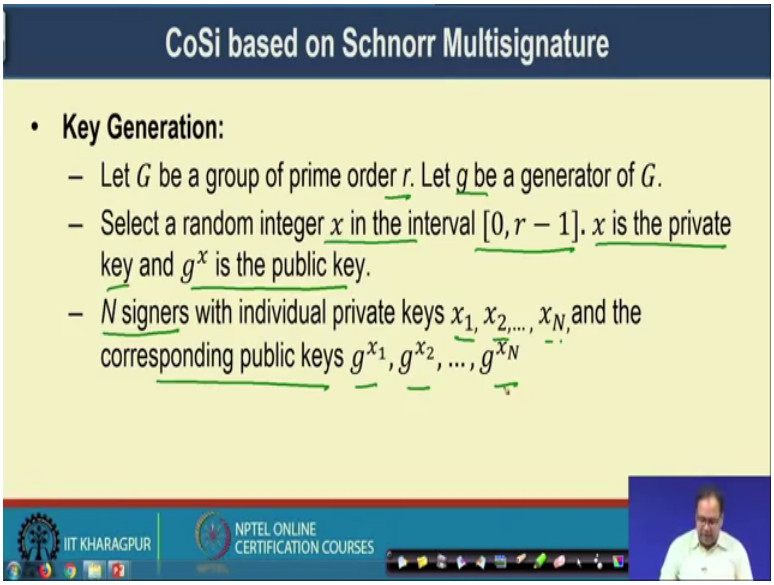
So, let us take an example to explain this. So, assume that you are having a group of multiplication say multiplication mod 5. So, this is the operation of multiplication mod 5. And say the generator of the group is assuming that it is a prime number; the generator of this group is say I am taking a random number 3. Now what are the members of this group? So, this will generate a cyclic group.

So, the member of this group will be 3 mod 5, then 3 cross 3 mod 5, then 3 cross 3 cross 3 mod 5, that means and so on. That means, you can say that this group will contain the element 3 to the power i mod 5 for i belongs to i belongs to say the set of natural numbers. Now in this case, you are getting the group element as say if I equal to 0, then you are getting it as 0, if I equal to 1, you are getting it 3. If I equal to 2, then you are getting 3 cross 3; that means, 9 mod 5 that means, 4. So, that way you can generating the element of this group. So, 3 is a generator of this particular group.

Now, we say that the discrete logarithmic problem on group is difficult. So, what is the discrete logarithmic problem for a group? Say you are giving a equation G to the power x mod. So, if I take this example mod 5, G to the power x mod 5, now assume that you know G, you know 5. And I am saying that this is having some value h. So, given this particular equation it is difficult to find the value of x if you know the generator the operator; that means, the module of 5 and the value of h. So, that is the discrete logarithmic problem on a group of prime order. So, in case of a group of a prime order,

we believe that this kind of discrete logarithmic problem is hard. So, this Schnorr multi signature it relies on this discrete logarithmic problem on a group of prime order ok.

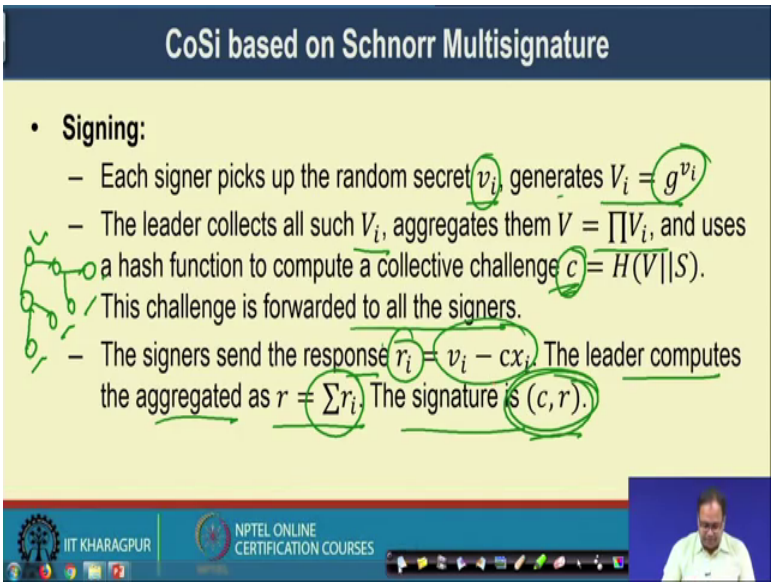
(Refer Slide Time: 21:15)



So, this is the basic idea of Schnorr multi signature to generating this collective signature over a group of prime order. So, assume that G is a group of prime order r and small g is a generator of G. So, if small g is a generator of G; that means, using g to the power g mod r g to the power 1 mod r, that way you can generate all the elements of that particular group.

Now, you select a random integer x in the interval 0 to r minus 1 and here in this Schnorr multi signature scheme x is the private key, and g to the power x is the public key. Now we have now multiple singers who was signing it all together. So, I have n different signers. So, this n different signers they have their individual private keys, x 1 x 2 up to x n, and the corresponding public keys G to the power x 1 G to the x 2 and up to G to the power x n. So, here these are the public keys and the private keys for the individual singers. So, it follows the basic digital signature principle that I will sign with my private key. And the sign can be validated with my public key which will be available to others.

(Refer Slide Time: 22:41)



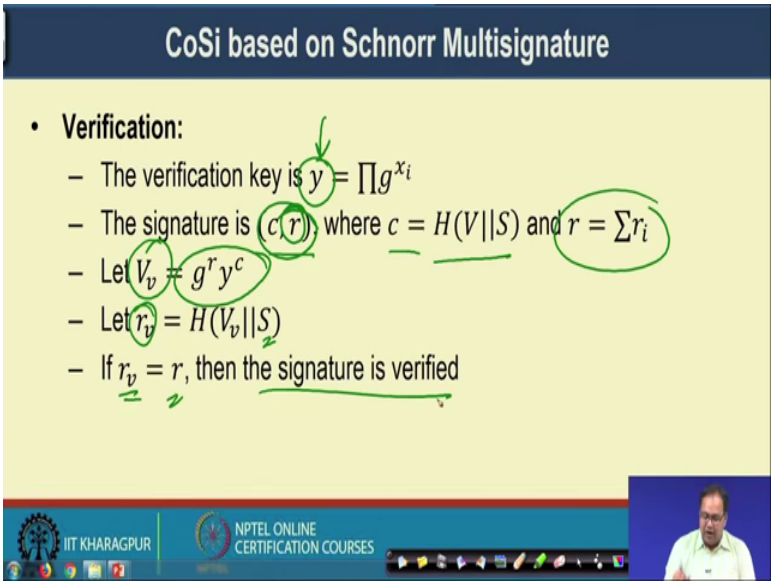
Well, so, the signing procedure is something like this. So, the signing procedure is that each signer. It picks up a random secret v i and generates this capital V i which is equal to g to the power v i, then the leader it collects all such v i from all the children. So, if you look into the remember this tree structure so, the root of the tree is the leader, and this individual nodes are the signers. So, everyone so that thing is that the protocol works in this way; that the leader get some random secret v i generates and every node generates this capital V i g to the power v i that will be generated by every random nodes after selecting these random secret. And here this v i is basically the private key and g to the power v i is the public key.

So, the leader it collects all such capital V i aggregates them as the product of capital V i. So, the leader collects all this v i and finds out this aggregate with the product and uses a hash function to compute the collective challenge called c. So, c would be hash of this v appended some signature s or some predefined c r which is s. So, these challenges forwarded to all the signers.

Now, all the signers they send the response r i, r i is equal to v i minus c of x i and the leader it computes the aggregate by combining all these r i. So, this r i leader compute the value by combining all these r is together which is equal to r. Now the combined signature is now c and r. So, the c that we have got from the challenge, and the response where every node has found out this value of v i that becomes that the aggregate of that

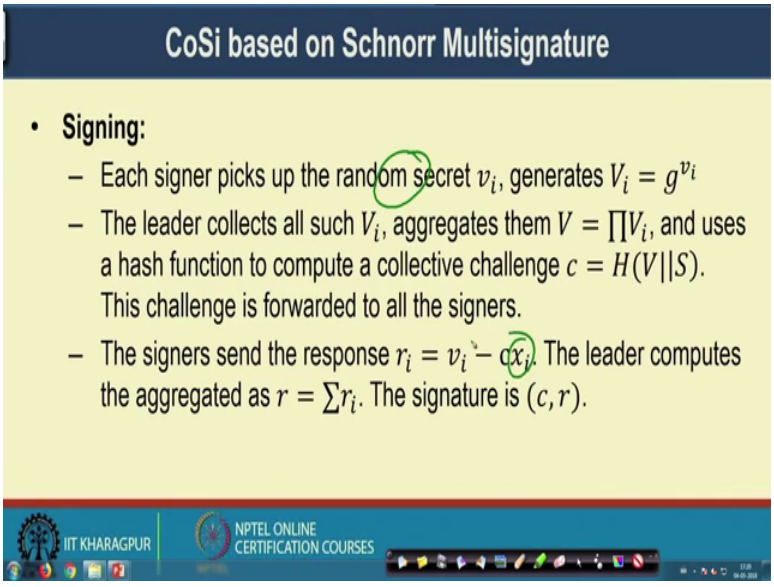
particular response; that means, the sum of that becomes my signature. So, this is the signature which is there.

(Refer Slide Time: 25:07)



Now, verification procedure is something like this. So, you generate the verification key y. So, if you remember G to the power x y G to the power x i is the G to the power x i is the x i is the public key. So, earlier we had this has been encrypted or the signature has been generated with this x i, which is the private key, and which is the private key for individual nodes.

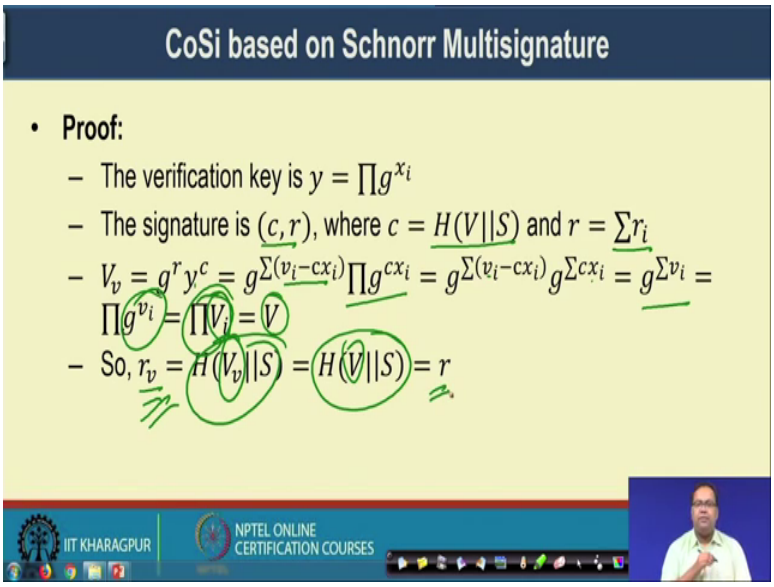
(Refer Slide Time: 25:31)



Now, we have generated we have generated this G to the power x i. So, this G to the power x i is the public key. So, that is aggregated verification key. So, the signature is c r so, where c is h v appended s, and r is the sum of all the r i that we have looked in the previous day. So, every node it computes a verifier if it wants to verify the signature. It computes this v v which is G to the power r y to the power c.

So, that is the parameter it computes with the help of this y and c r. So, the signature is available and y is available to the verifier. It can compute this caps v subscript small v. And assume that r v is equal to hash of this v appended the c rth that is there some s. So, this s you can just think of the message which is coming for verification. Now, if this r v becomes equal to r, which was there is a part of the signature, then the signature is verified. Now let us look a proof of this.

(Refer Slide Time: 26:53)



So, the proof work in this way. So, you have this aggregated verification key which is y product of G to the power x i. Now the signature is c r, where c is h v appended the c rth document. And r is the sum of all the r y. Now you are computing capital V subscript v which is equal to g to the power r y to the power c.

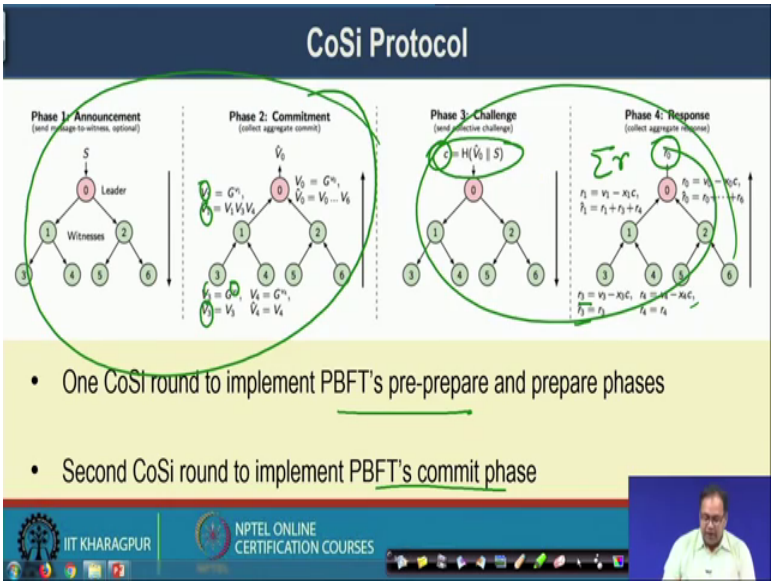
Now, r is if you remember r is sum of all the r i and r i was so, in the encryption procedure r i was v i minus c of x i; where x i is the private key for all the individual signers now. So, r i is v i minus c of x i. So, you put the value of r i here. Now this particular quantity and y to the power c y is product of G to the power x i. So, that to the

power c, it comes to the product of G to the power c x i. Now you just rearrange this term; that means, product of G to the power c x i, it becomes G to the power sum of all this c x i.

Now, this quantity, it becomes equal to G to the power sum of v i; so, this plus this. So, this c x i sum of c x i gets cancelled up and it become G to the power sum of v i, which is again equivalent to product of G to the power individual v i. So, this product of G to the power v i is equal to product of G to the power v i means caps v i. So, this is equal to my v.

Now, whenever you are calculating r v. So, it is v subscript v becomes equal to v. So, the hash value that you are getting here, that should be equal to the hash value that is you are getting here. So, r v should be equal to r if the signature is correct. So, that way you can verify the signature using Schnorr multi signatures.

(Refer Slide Time: 28:57)

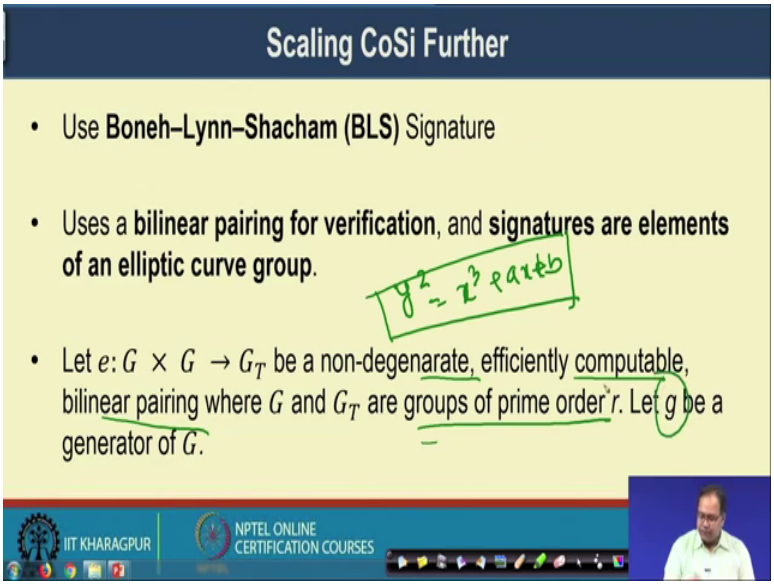


So now as we have mentioned that this 3 round of PBFT, you can simulate using 2 round of CoSi protocol. So, here are the 2 rounds of CoSi protocol. So, in the first round, the leader is making an announcement. So, this leader is so, you want to sign this document s. So, the leader is making an announcement sharing this s to all the witnesses. Now the secondth means the upward rounds. So, every round means one downward message propagation and one upward response propagation.

So, here during the upward response propagation, you are computing these values of v's the individual individual small v and the caps V, then the third in the in the second round; that means, the phase in phase 3 you are sending the collective challenge. So, the leader computes the challenge c, sends it to all the nodes. And then all the nodes finds out the response and sends it to the leader. So, this c the challenge and the collective response that mean the sum of all this r becomes my signature. So, that way one CoSi round is used to implement the PBFTs pre prepare and the prepared phase.

So, here everyone is getting the values of v 1 and the capital V 1. And the second CoSi round this to implementing the PBFTs commit phase; where you are sending a challenge and everyone is generating the response out of this challenge.

(Refer Slide Time: 30:29)

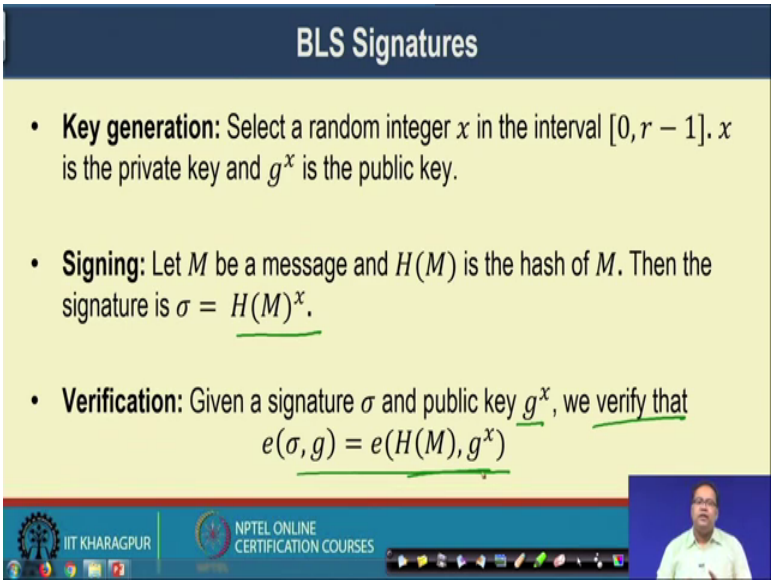


So, that is the basic CoSi protocol using the Schnorr multi signature scheme. There is another way you can scale up the CoSi protocol further using Boneh Lynn Shacham signature so, BLS signature. So, this BLS signature it uses a bilinear pairing for verification. And these signatures are the elements of an elliptic curve group.

So, in a elliptic curve group an elliptic curve is something like y square equal to x raise cube plus a x plus b. So, that means, the elements of the group that you are generating, you are generating it from an elliptic curve following an elliptic curve equation. So, that is used for the cryptography group formation purpose.

Now, assume that this e G cross G to G T is a non-degenerate efficiently computable bilinear pairing; where G and G T are the groups of prime order r. And assume in this groups of prime order you have small g as a generator of G. Now in this case the in BLS signature you consider an instance of this computational Diffie Hellman problem where you have the value of G g to the x G to the power y. So, the CDH problems tells you that the pairing function e, it does not help you to compute G to the power x y the solution of the CDH problem. So, once you have the value of G and the value of G to the power x and G to the power y, from there you will not be able to compute G to the power x y.

(Refer Slide Time: 32:03)

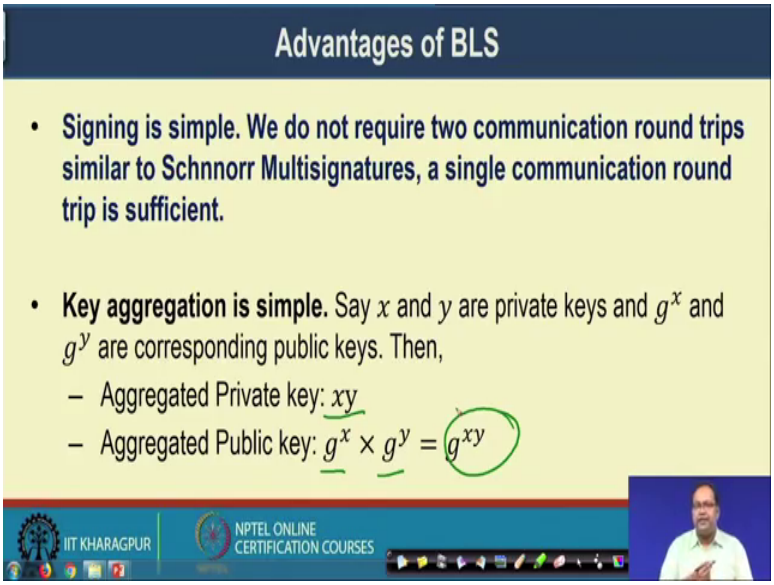


So, that is the CDH problem which we use in case of BLS signature. So, in case of BLS signature they key generation and the signing is a very simple. So, the key generation says that you select a random integer x in the interval 0 to r minus 1; where you are having a group of prime order r, and here your x is the private key and G to the power x is the public key. Now, assume that M is a message that you want to sign and H M is the hash of M. Then you can simply sign it by H M to the power x which is your signature.

Verification is through this bilinear pairing in elliptic curve. So, given a signature sigma and the public key G to the power x, you can verify if the pairing with sigma and G is equal to the pairing of H M and G to the power x; if this 2 becomes equal, that means the signature gets verified. So, I am not going to the detail proof of this BLS signature you

can look into the standard cryptographic book or the net to find out the correctness proof of BLS signature.

(Refer Slide Time: 33:13)



But the interesting thing in BLS signature is that first of all the signing is very simple. So, you do not require two communication round trip similar to this challenge and response in Schnorr multi signature. So, a single communication round trip is sufficient. You just send the message and everyone will be able to sign it. And the key aggregation is very simple. So, the key aggregation in BLS work in this way say x and y are the private keys, and G to the power x and G to the power y are the corresponding public key, then your aggregated private key will be xy and your aggregated public key will be G to the power x into G to the power y, which is G to the power xy.

So, that way getting the aggregated key is very simple in case of BLS. And also you do not require two communication round trips for Schnorr multi signature. So, these are the concept of the CoSi protocol; where by utilising this Schnorr multi signature or the BLS based signature scheme, you can have a collective way of signing a document through which you can verify the correctness of a of a particular statement or a particular document.

So, in the next class we will discuss that how this CoSi is incorporated in case of Byzcoin by solving the problem which was there in Bitcoin NG.

So, thank you all for attending this class.