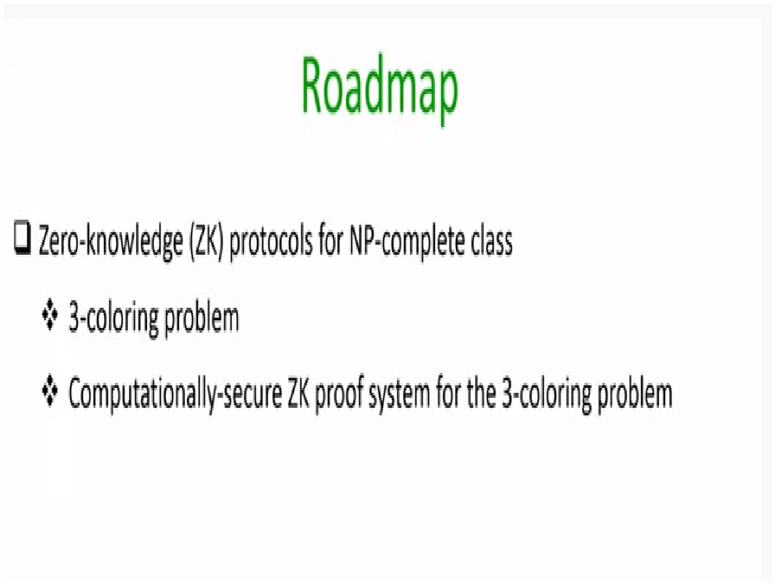
Foundations of Cryptography

Dr. Ashish Choudhury
Department of Computer Science
Indian Institute of Science – Bangalore

Lecture – 58
Zero-knowledge Protocols Part II

Hello everyone, welcome to this lecture. Just a quick recap. In the last lecture we had started our discussion on zero-knowledge protocols. So in this lecture we will continue our discussion on zero-knowledge protocols specifically we will introduce zero-knowledge protocols for NP-complete class.

(Refer Slide Time: 00:44)

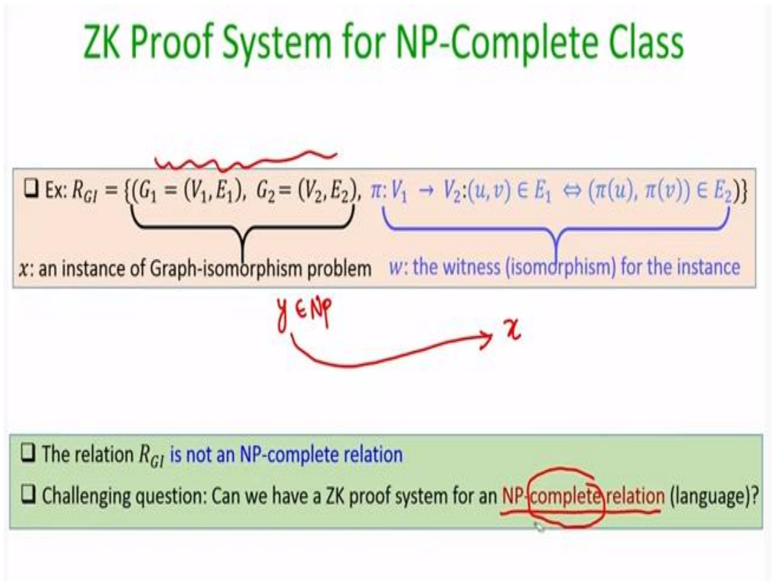


Roadmap

- Zero-knowledge (ZK) protocols for NP-complete class
  - 3-coloring problem
  - Computationally-secure ZK proof system for the 3-coloring problem

For that we will introduce the 3-coloring problem and we will see a computationally secure zero-knowledge proof system for 3-coloring problem.

(Refer Slide Time: 00:51)



ZK Proof System for NP-Complete Class

- Ex: $R_{GI} = \{(\underbrace{G_1 = (V_1, E_1),\ G_2 = (V_2, E_2)}_{x\text{: an instance of Graph-isomorphism problem}},\ \underbrace{\pi: V_1 \to V_2: (u, v) \in E_1 \Leftrightarrow (\pi(u), \pi(v)) \in E_2}_{w\text{: the witness (isomorphism) for the instance}})\}$

- The relation R_{GI} is not an NP-complete relation
- Challenging question: Can we have a ZK proof system for an NP-complete relation (language)?

So, recall that in the last lecture we have seen that how exactly we can specify a relationship. For instance, if you recall the relationship for graph-isomorphism problem then the relationship will consist of (x, w) pairs where x is a problem instance. So in this example if we consider a graph-isomorphism problem than the x instance is basically the publicly known description of 2 graphs and the witness component corresponding to this x instance will be the isomorphism mapping between the vertex set of the first graph to the vertex set of the second graph.

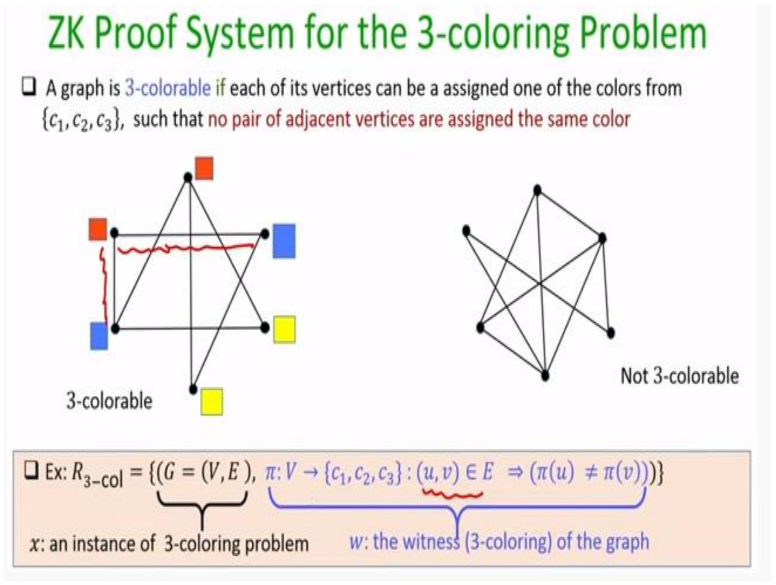
And indeed if we have an x or problem instance where the 2 graphs are isomorphic then we should have a corresponding witness w. In the last lecture, we have seen that how we can come up with a zero-knowledge proof system which allows the prover to show whether an instance x has a corresponding witness w available with the prover and not without revealing anything about the witness w.

But it turns out that the relation graph-isomorphism is not an NP complete relation and next challenging question is can we have a zero-knowledge proofs system for any NP complete relation? So, for people who might be wondering what exactly is NP-complete problem or NP-complete relation is. A problem x is called an NP-complete problem, if given a witness w, we can verify whether indeed the witness w is a right witness for the problem instance x and in polynomial amount of time. Specifically by performing non-deterministic computation for polynomial amount of time. That is the first requirement. The reason we call such problems as NP-complete, the completeness aspect here denotes that if we have any other problem y belonging to the class NP then that problem instance y can be reduced to an instance of the problem x in polynomial amount of time. In that sense this problem x will be called as an NP complete relation.

That means if you have a solution to solve problem instance x, that means if you can find out witnesses for problem instance x in polynomial amount of time, then by just using the reduction of problem instances of y to the problem instance of x, you can also get solutions for your problem instances y. So that is a rough definition of NP complete relation.

So, we are now interested to see whether we can come up with a zero-knowledge proofs system for any relation which is NP complete. So, it turns out that the graph- isomorphism is not an NP complete relation.

(Refer Slide Time: 03:35)



ZK Proof System for the 3-coloring Problem

- A graph is 3-colorable if each of its vertices can be a assigned one of the colors from $\{c_1, c_2, c_3\}$, such that no pair of adjacent vertices are assigned the same color

- Ex: $R_{3\text{-col}} = \{(\underbrace{G = (V, E)}_{x\text{: an instance of 3-coloring problem}}, \underbrace{\pi: V \rightarrow \{c_1, c_2, c_3\} : (u, v) \in E \Rightarrow (\pi(u) \neq \pi(v))}_{w\text{: the witness (3-coloring) of the graph}})\}$

So, what we are going to now do is we are going to see a zero-knowledge proof system for another computational problem which we call as 3-coloring problem which is a well-known NP complete problem. So let us first see what we mean by a 3-colorable graph? We say a given graph with n-vertices is 3-colorable if each of its vertices can be assigned one of the colors from publicly known colors c_1, c_2, c_3 such that no pair of adjacent vertices are assigned the same color. That means we have to color the vertices in such a way that every pair every the endpoints of every edge should have different colors.

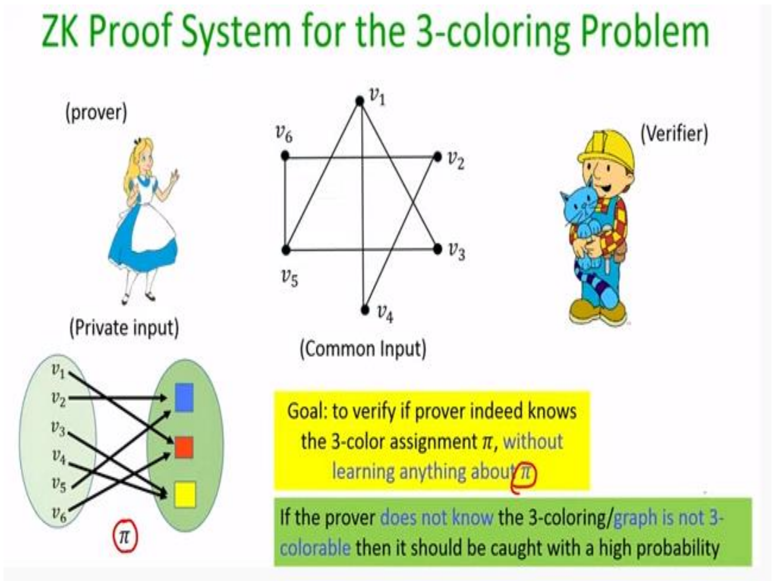
If you consider this graph for instance, then this is a 3-colorable graph. So, for instance if my {c_1, c_2, c_3} are {red, blue, yellow} then we can color the vertices in this graph using these 3 colors by one of these assignments by assigning these colors to the respective vertices. And now you can see that no 2 adjacent vertices namely no 2 adjacent vertices are assigned the same color here.

For instance, if I consider these 2 vertices, they are adjacent in the sense that they are the end points of a single edge say same edge and they are having different colors. In the same way if I consider this edge the end points are getting different colors and so on. On the other hand, if I consider the graph on your right-hand side then it is not 3-colorable. That means it is not at all possible to color all the vertices of this graph with just 3 colors satisfying the condition that no pair of adjacent vertices are assigned the same color.

The 3-coloring problem or the 3-coloring relation is a well-known NP complete relation and the (x, w) entry in the 3-coloring relation will look like this. So, the x instance will be the public description of a graph namely the number of vertices and the vertex set, and the edge set of the graph will be publicly known.

If indeed this x instance namely this graph is 3-colorable then the corresponding witness w will be the mapping of or the assignment of the color {c_1, c_2, c_3} to the vertex set which I call a π. And the coloring the witness π should satisfy the restriction that if the edge (u, v) belongs to the edge set then the color assigned to the node u and the color assigned to the node v should be different. If indeed it is possible to come up with such an assignment of color π for the given problem instance x then (x, w) will be considered as a valid entry or satisfying the relationship 3-coloring. If we cannot find a witness w, then corresponding to a given x then that x will not be present in the 3-coloring relation.

(Refer Slide Time: 06:29)



ZK Proof System for the 3-coloring Problem

Goal: to verify if prover indeed knows the 3-color assignment π, without learning anything about π

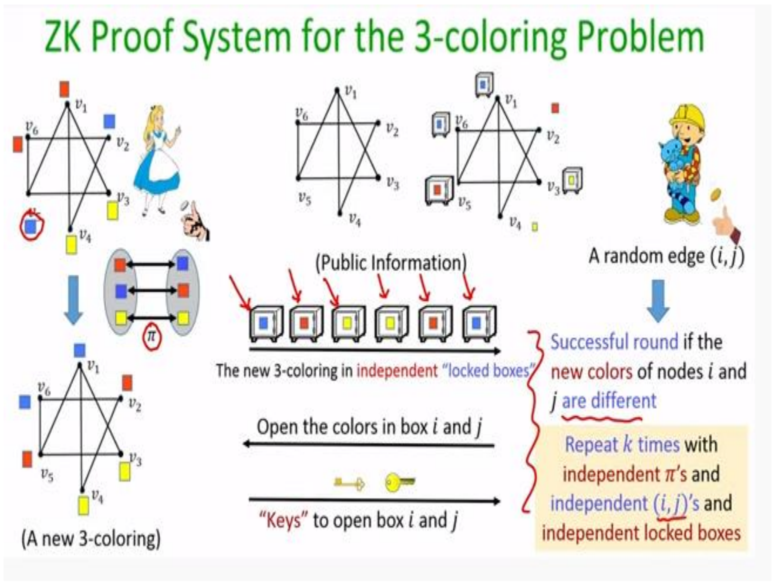
If the prover does not know the 3-coloring/graph is not 3-colorable then it should be caught with a high probability

So now let us see as you know knowledge proof system for the 3-coloring problem. So, imagine a Alice is the prover and Bob is the verifier. The common input for both Alice and Bob is the description of a graph. Say, the private input for the prover namely Alice here is a 3-coloring of the publicly known graph namely an assignment π mapping the vertex set to the color set {c_1, c_2, c_3}and what Alice wants to prove to Bob that indeed the given graph is 3-colorable and she has an assignment π available with her.

So, the goal here is to come up with a zero-knowledge proofs system which should convince Bob that indeed Alice knows the corresponding mapping π without revealing anything about the actual mapping π. At the same time the zero-knowledge proofs system should ensure that if prover does not know the 3-coloring of the given graph or if the graph is not 3-colorable at the first place then with very high probability while giving the proof, Alice should be caught by Bob.

Recall, that this is your soundness property, the second requirement. And the first property namely Bob should not learn anything about the 3-coloring is the zero-knowledge property.

(Refer Slide Time: 07:56)


ZK Proof System for the 3-coloring Problem
v_1
v_2
v_3
v_4
v_5
v_6
π
(Public Information)
A random edge (i, j)
The new 3-coloring in independent "locked boxes"
Successful round if the new colors of nodes i and j are different
Open the colors in box i and j
Repeat k times with independent π's and independent (i, j)'s and independent locked boxes
"Keys" to open box i and j
(A new 3-coloring)

So now let us see how exactly the zero-knowledge proof system for the 3-coloring problem will look like. Alice has the private 3-coloring available with her. So, what she does is she randomly permutes the color that she has available with her namely see she has the secret mapping π sorry she has the original coloring of the graph. So, what she does is she creates a random permutation of the color set.

For instance, she can say create a permutation we are red gets mapped to blue, blue gets mapped to red and the third color remains as it is as per the permutation. Basically, now what she is doing is she is creating a new 3-coloring of the graph that is available with Bob with respect to the new mapped colors. So wherever in the original graph whichever vertices were colored with the red color those vertices in the new coloring will be assigned blue color because the red color gets mapped to blue color.

In the same way in the old coloring whichever vertices were assigned the blue color those vertices in the new 3-coloring will be assigned a red color and so on. All this Alice is doing at her end. Now once Alice computes the new 3-coloring of the graph what she does is she keeps the new assigned colors of the respective vertices in a locked box. And here locked boxes and quote unquote. We will later see what exactly are the properties we require from this locked boxes and how do we instantiate it using cryptographic primitive.

So, on a very high level what these locked boxes means is that seems in this example we have 6 vertices. So what Alice is doing is whatever is the new color assigned to the vertex one that is kept inside this locked box where the locked is available with Alice. Its locked box in the sense that without having access to the key, Bob cannot open the first box and see what is the new color of the first vertex. In the same way the new color of the second vertex is in the second lockbox the new color of the third vertex is in that third locked box and so on.

Bob right now cannot see the colors that are available in the locked boxes. So from the viewpoint of the Bob if indeed Alice knows the original 3-coloring then from the viewpoint of the Bob, Bob will feel as if he is now seeing a new 3-coloring of the same publicly known existing graph with the exception that he actually do not know what are these exact colors now because all those new colors are in the locked box.

So, this first round of message is like a commitment from Alice to Bob. That means she is saying that “okay! I have now computed a new 3-coloring. I will not show you the new 3-coloring. Those new 3-colorings are actually available in these locked boxes.” That is the commitment from my side as per the zero-knowledge proof system. Now once Bob receives the commitment of Alice, what Bob does is it creates a challenge for Alice. The challenge is basically a random edge (i, j) from the graph.

Remember Alice will not be knowing what exactly is the random edge that Bob will challenge when Alice is committing the new colors in the locked boxes. So once Bob picks the random edge, he challenges Alice that “you please open the new colors in the box i and the box j. I want to check whether indeed they are different colors or not. Because if at all you know the original 3-coloring then as per your mapping the new coloring should also be a 3-coloring.”

And hence since i and j are adjacent nodes the new color of the node i and a new color of the node j should ideally be different so that is what Bob is challenging Alice to show. And to respond to Bob's challenge what Alice reveals is it reveals is the keys for the box i and the box j. Now I need another property from the locked box here. So remember, one of the properties of the locked boxes is that whatever is kept inside the locked box Bob cannot see its content until and unless he gets access to the keys of the box.

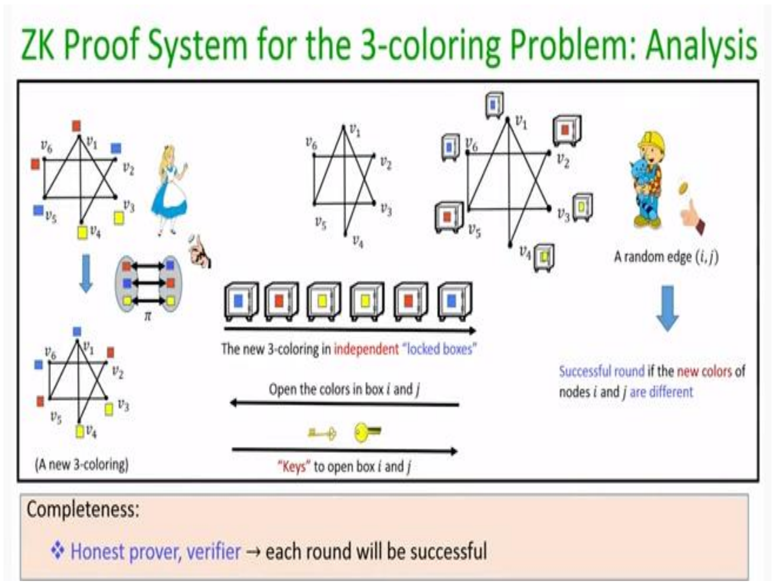
The second property that I require from this locked boxes is that once Alice has kept something inside the locked box and if later, she is asked to reveal the contents of any of the boxes she later cannot change the content that she has already kept inside that particular box. So in this case Bob is challenging Alice to open the box i and box j and Alice is forced to give the keys for the box i and box j. As per the properties of the lockbox whatever she has committed inside the ith box and the jth box that she is not allowed to change.

Now Bob will verify. Indeed Bob will take the keys for ith box and the jth box, will open those boxes so they are the new colors with respect to the new 3-coloring for the graph. Bob will consider it to be a successful round if it finds that the new colors of the node i and the node j are different which should ideally be the case. If this is not the case, then the round is unsuccessful and Bob stops the protocol here itself. So this is how one round of the zero-knowledge proofs system works.

Now to boost the confidence in this proof what Bob can do is Bob can repeat this process k number of times independently where in each round Alice can use a different 3-coloring with respect to the old 3-coloring techniques. That means every round she will be picking an independent π mapping the existing 3-coloring to a new 3-coloring and independently Bob will be picking fresh challenges (i, j) in every round.

That means it will not be the case that (i, j) which is picked as the challenge edge by the Bob will remain the same in all the rounds. They will be picked independently, and Alice will not be knowing anything about the challenges for the subsequent round in advance and if all this k rounds are successful Bob will consider that indeed Alice knows the actual or the original 3-coloring for the existing graph.

(Refer Slide Time: 14:18)



ZK Proof System for the 3-coloring Problem: Analysis

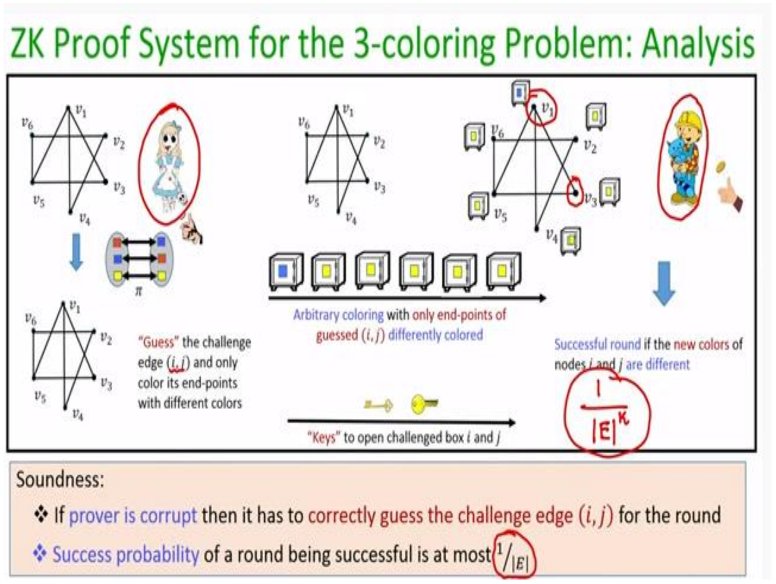
v_1 v_2 v_3 v_4 v_5 v_6
π
(A new 3-coloring)
The new 3-coloring in independent "locked boxes"
Open the colors in box i and j
"Keys" to open box i and j
A random edge (i, j)
Successful round if the new colors of nodes i and j are different

Completeness:

- Honest prover, verifier → each round will be successful

So that is a zero-knowledge proof system. Now let us try to do the analysis for the zero-knowledge proofs system whether it satisfies the requirement of correctness, soundness and zero-knowledge. Correctness or completeness, so remember completeness property means that if Alice and Bob are honest and if indeed Alice has a witness for the graph namely she has the original 3-coloring then with high probability the proof should go through and Bob should be convinced. It is easy to see that if indeed Alice knows the original 3-coloring then indeed in all the rounds she will be able to successfully convince Bob. She will not fail and hence each round will be successful, and Bob will accept the proof.

(Refer Slide Time: 15:03)



ZK Proof System for the 3-coloring Problem: Analysis

v_1 v_2 v_3 v_4 v_5 v_6
π
"Guess" the challenge edge (i, j) and only color its end-points with different colors
Arbitrary coloring with only end-points of guessed (i, j) differently colored
"Keys" to open challenged box i and j
Successful round if the new colors of nodes i and j are different
$\frac{1}{|E|^k}$

Soundness:

- If prover is corrupt then it has to correctly guess the challenge edge (i, j) for the round
- Success probability of a round being successful is at most $1/|E|$

Now let us consider the soundness property. So, remember for soundness property we have to consider the case when the prover is corrupt. Prover is corrupt in the sense that either the graph is not 3-colorable, or it might be the case that the graph is 3-colorable, but Alice does not know what exactly is the 3-coloring of the original graph. For simplicity and without loss of generality assume that the graph is not 3-colorable.

So now let us analyze what is the probability that any round is successful with respect to a potentially corrupt Alice who does not know or who for a graph where the graph is not 3-colorable. It turns out that if Alice is corrupt and Bob is honest then the only way Alice can still successfully pass the round is when she can guess in advanced the edge (i, j) which is going to be picked as the challenge by Bob.

Because if the graph is not 3-colorable then Alice cannot create any new 3-coloring for the graph because the graph is not 3-colorable at the first place. Then the only way Alice can win is that she can guess. She can pretend in her mind that this might be the edge (i, j) which Bob can challenge me to show. So, what she can do is she can assign arbitrary coloring to the end points of the graph. In fact, she can assign same colors to all the nodes in the graph except the end points i and j.

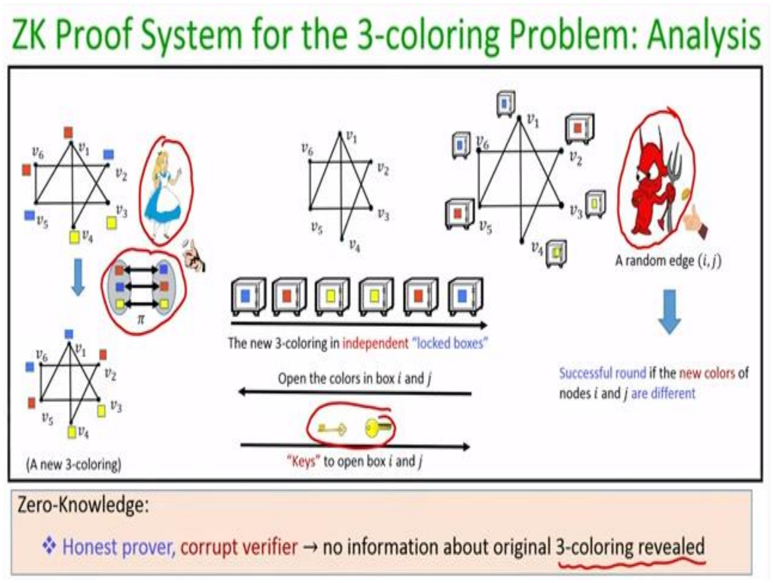
Because Alice can just guess that this might be the edge which Bob can ask me to open. What Alice can do for instance she can guess that i might be a say 1 and j might be equal to anything say semantics number 3? So, for instance she can imagine that she might be challenged to show the new colors for the graph for the edge (v_1, v_3) or for the end points v1 and v3. What she can do is for the first locked box she can keep the color blue and the third locked box she can keep the color yellow and then in all the other boxes she can just keep the same colors.

And if indeed she is lucky, then it might be the case that she is indeed asked to open the first locked box and the third locked box. Then, she will be successfully showing Bob that "Hey! I have assigned different colors to the node number v_1 and node number v_3. But what is the success probability of Alice guessing in advance that what will be the random (i, j) which Bob will challenge Alice to open.

The probability that Alice is successfully able to guess the random challenge (i, j) is nothing but 1 over the size of the edge set. Approximately the size of edge set in the worst case can be n^2. That means success probability of the round being successful is bounded by 1 over the edge set namely $1/ n^2$. Remember that there are k such rounds. That means the only way Alice without even knowing the 3-coloring of the graph can successfully pass all the k rounds is when for each of the k-rounds she can guess correctly in advanced that challenge (i, j) which Bob will be asking in each round.

And the success probability of guessing that in one round is $1/E$. The probability that she can do it in all the k rounds is nothing but $1/|E|^k$. By setting k to be sufficiently large it can be ensured that this quantity $1/|E|^k$ becomes very small. Hence definitively in one of the k rounds Alice will get caught. If she gets caught in any of these k rounds, Bob will suspend the proof system and the claim of Alice will be rejected. So that proves the soundness property.

(Refer Slide Time: 19:07)



ZK Proof System for the 3-coloring Problem: Analysis

v_1 v_2 v_3 v_4 v_5 v_6
π
(A new 3-coloring)
The new 3-coloring in independent "locked boxes"
Open the colors in box i and j
"Keys" to open box i and j
A random edge (i, j)
Successful round if the new colors of nodes i and j are different

Zero-Knowledge:

❖ Honest prover, corrupt verifier → no information about original 3-coloring revealed

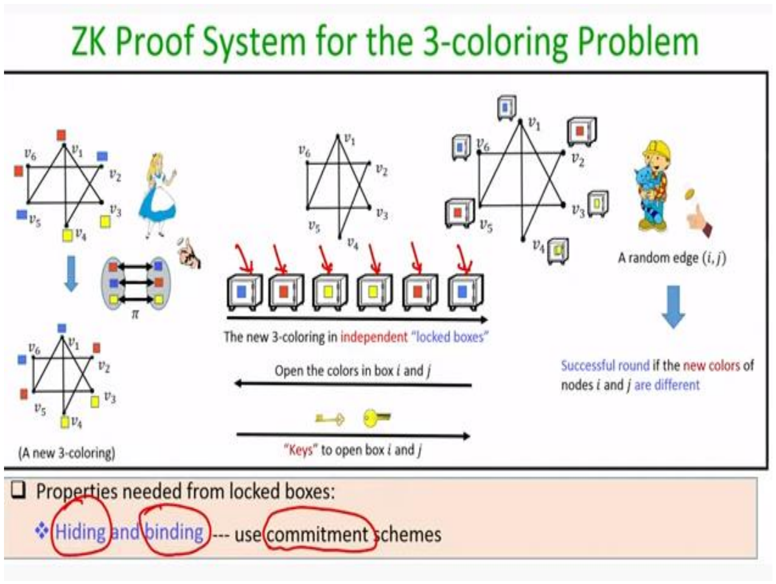
Now let us analyze a zero-knowledge property and remember for the zero-knowledge property we have to consider the case when our prover namely Alice is honest and indeed, she has a witness which she wants to hide from a malicious verifier. So in this case the verifier is the corrupt guy and the goal of the verifier is to try to learn about the original 3-coloring. It turns out that even if the verifier is corrupt, it learns absolutely no information about the original 3-coloring.

This is because what he is seeing in the first round. He is seeing the new 3-coloring but not the exact new 3-coloring, but rather the verifier is seeing the new colors assigned to the vertices in the graph all of which are kept in the locked boxes. It is only a pair of locked boxes which is opened by Alice in response to the challenge thrown by our verifier.

So even if the verifier sees the color of the node i and the node j, they are the new 3-coloring. They correspond to the new 3-coloring and it learns only the colors for the ith node and the jth node but not the entire new 3-coloring. Remember in each of the rounds, Alice is picking a fresh independently chosen π. That way she is randomly permuting the existing 3-coloring and creating a new 3-coloring.

So that means the new 3-coloring in the first round will be independent of the new 3-coloring in the second round and like this the new 3-coloring in the kth round will be independent of all the new 3-colorings in the previous round. That means in each of the round, verifier is just going to learn that okay I will be seeing the new colors of the node i and node j which I know are going to be distinct. And that is why it does not reveal anything about the original 3-coloring which was available with Alice. So that proves the zero-knowledge property.

(Refer Slide Time: 21:01)



ZK Proof System for the 3-coloring Problem

v_1
v_2
v_3
v_4
v_5
v_6
π
The new 3-coloring in independent "locked boxes"
Open the colors in box i and j
"Keys" to open box i and j
(A new 3-coloring)
A random edge (i, j)
Successful round if the new colors of nodes i and j are different

- Properties needed from locked boxes:
  - Hiding and binding --- use commitment schemes

Now coming back to the question that what are the properties we need from the locked boxes? As I said we need 2 properties. We need basically the hiding property. Namely if prover is honest, if

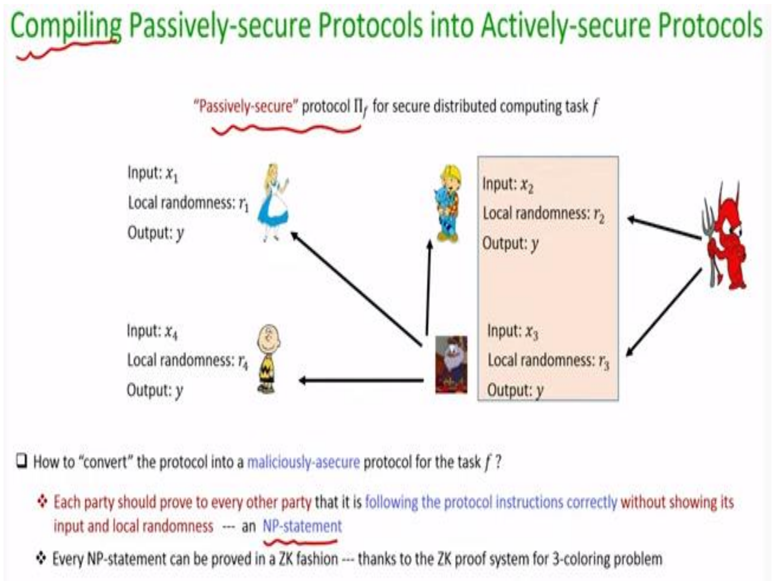
Alice is honest and if she has kept some contents inside the box, then until and unless the keys for those boxes are given to Bob, he cannot open and see the contents that are kept inside the box. So that is what I mean by the hiding property here.

Binding property means if Alice is corrupt then it should not be the case that she put something inside the box but when she is supposed to open the box, she can turn it or she can change it into any new content. So that is a binding property and now we very well know how to instantiate this locked box both with both these 2 properties.

Basically, we can use a commitment scheme so that means what Alice has to do is in each round, she has to compute a new 3-coloring and a new color for the vertices. She has to commit by using any commitment scheme. When Bob challenges to open the new colors of ith node and the jth node, Alice has to give the opening information corresponding to the ith commitment and the jth commitment.

So, this proves that we now have a 3-coloring. We now have a zero-knowledge proof system for the 3-coloring problem and its well known that the 3-coloring problem is an NP-complete problem that means now we have a zero-knowledge proof system for any NP relation.

(Refer Slide Time: 22:31)



Compiling Passively-secure Protocols into Actively-secure Protocols

"Passively-secure" protocol Π_f for secure distributed computing task f

- How to "convert" the protocol into a maliciously-asecure protocol for the task f ?
  - Each party should prove to every other party that it is following the protocol instructions correctly without showing its input and local randomness --- an NP-statement
  - Every NP-statement can be proved in a ZK fashion --- thanks to the ZK proof system for 3-coloring problem

Now let us see the power of the zero-knowledge proof system. What we can now do is we can see a very nice framework namely a compiler which can compile any passively secure protocol into actively secure protocol. Imagine, you have a distributed computing task say f, it can be any abstract computational task. It could be say, for example a task involving multiple parties 2 parties, 3 parties or say 4 parties or any n number of parties where each party have some input say x_1, x_2, x_3, x_4.

I am taking the case where I have 4 parties. The goal of the parties is basically to compute $f(x_1, x_2, x_3, x_4)$ and in such a way that even if there are some bad guys in the system, they do not learn anything about the x inputs of the good guys other than what they can learn from their own input and the function output. This is a very abstract problem. This problem you also call as multiparty computation problem.

The way any multi-party computation protocol will work as follows: the parties will have their own inputs and they will choose to some local randomness say r_1, r_2, r_3, r_4 respectively and then they will interact with each other as per the instructions of this protocol π_f. At the end they will obtain the function output y where $y = f(x_1, x_2, x_3, x_4)$ and for the moment imagine that this protocol π_f is passively secure.

It is passively secure in the sense that if even if there is an adversary who can control or who can see the input and the local randomness of some fraction of these n parties and whatever messages they have exchanged during the protocol, by seeing their inputs the output and the messages that they have received and they have sent during the protocol, the bad guy does not learn anything additional about the inputs of the good guys. So that is what I mean by saying that this protocol π_f is passively secure.

Now imagine I want to compile this protocol. Compiling this protocol in the sense I want to retain the protocol π_f and I want to ensure that the protocol remains secure even if there is an active adversary or a malicious adversary. Active adversary, that means I want to compile this protocol into a maliciously secure portal sorry for the typo it should be maliciously secure protocol.

So what I mean by malicious security here is that even if the bad guys who are under the control of their adversary try to deviate from the instructions of the protocol, they should not learn anything about the inputs of the good guy. I do not want to design a new protocol or a fresh protocol. I just want to retain the protocol π_f which is guaranteed to be secure against the passive adversary.

So how can we compile the passively secure protocol into a maliciously secure protocol? That is a question that we want to know the answer. Now we want to use a zero-knowledge proof system here. So, it turns out that a generic way to convert the passively secure protocol into a maliciously secure protocol is as follows: if each party proves to every other party that it is indeed following the protocol instructions correctly, then in the presence of a malicious adversary, the protocol π_f will be achieving its task.

Now the question is what we mean by saying that a party proving to other party that indeed it is following the protocol instruction correctly. By that I mean that each party has to prove to every other party that the messages that they are sending are indeed with respect to their randomness and their input as per the instructions given by the protocol π_f. How can the other parties verify whether indeed each party is following its protocol instruction or not?
Well it can check the messages that particular party is sending and the witness whether that party is sending or performing its action properly or not will be the parties input and the local randomness. For instance, in this case if Alice wants to verify whether indeed this third party is following his protocol instruction correctly or not? then one way of verifying that is Alice checks the messages which this third party is sending. Along with that if this third party shows his input x_3 and his randomness r_3 which he has used as part of this protocol π_f to Alice then indeed Alice can perform the action of this third party, because the description of the protocol is known what was not known was x_3 and r_3.

But now x_3 and r_3 is also given to Alice, so she can herself compute the messages which this third party is supposed to send as per the protocol π_f. If those messages match the messages that indeed this third party has sent or communicated during the real execution of the protocol, that1 proves that indeed this third party is following its step as per the protocol π_f. Like this every party can verify every other party's action whether they are performing their actions as per the protocol π_f, if that sending party reveals its input and local randomness.

But it turns out that the input and the local randomness of every party cannot be given to other parties because that is what ensures the security of the protocol π_f. If I learn the input and the randomness of every other party, then there is no way I can guarantee the security of the protocol π_f. So, it turns out that the statement which every party wants to prove to every other party namely I am following the protocol instruction is nothing but an instance of an NP statement.

The problem instances is the set of messages that I have sent, and I want to prove to you that corresponding to these messages I have some randomness and some input such that these messages are indeed computed consistently as per those input and the randomness as per the protocol instruction π_f. So, what each party now has to basically do to convince to other party that it is indeed following the protocol instruction it has to basically prove an NP statement.

And interestingly we now have a zero-knowledge proof system for the 3-coloring problem which is an NP-complete statement. Since it is NP-complete statement that means any instance of NP problem or any NP statement can be reduced to an instance of this 3-coloring problem. That means we can now use the zero-knowledge proof system for the 3-coloring problem. Each party can transform an instance of the NP statement namely that it is following the protocol instruction correctly into an instance of the 3-coloring problem and give us zero-knowledge proof for the existence of 3-coloring and convince to the other parties that indeed it is following the protocol instructions.

If the proof gets satisfied that means that gives the guarantee that every party is following the protocol instructions correctly. If the proof does not go through, we can simply stop the protocol there itself. So, in that sense the zero-knowledge proof system, it gives you a very powerful paradigm of compiling a passively secure protocol into a maliciously secure protocol. So that brings me to the end of this lecture.

Just to summarize in this lecture we have seen zero-knowledge proof system for the 3-coloring problem. And 3-coloring problem is a well-known NP-complete problem and we have seen that how using zero-knowledge proof system we can compile any passively secure protocol for any distributed computing task into a protocol which will be secure even against a malicious adversary. Thank you!