Notes for Lecture 24

Scribed by Milosh Drezgich, posted May 11, 2009

Summary

Today we introduce the notion of *zero knowledge proof* and design a zero knowledge protocol for the *graph isomorphism* problem.

1 Intuition

A *zero knowledge proof* is an interactive protocol between two parties, a *prover* and a *verifier*. Both parties have in input a *statement* that may or may not be true, for example, the description of a graph G and the statement that G is 3-colorable, or integers N, r and the statement that there is an integer x such that $x^2 \bmod N = r$. The goal of the prover is to *convince* the verifier that the statement is true, and, at the same time, make sure that *no information other than the truth of the statement* is leaked through the protocol.

A related concept, from the computational viewpoint, is that of a *zero knowledge proof of knowledge*, in which the two parties share an input to an NP-type problem, and the prover wants to convince the verifier that he, the prover, *knows a valid solution for the problem on that input*, while again making sure that no information leaks. For example, the common input may be a graph G, and the prover may want to prove that he knows a valid 3-coloring of G, or the common input may be N, r and the prover may want to prove that he knows an x such that $x^2 \bmod N = r$.

If a prover "proves knowledge" of a 3-coloring of a graph G, then he also proves the statement that G is 3-coloring; in general, a proof of knowledge is also a proof of the statement that the given instance admits a witness. In some cases, however, proving that an NP statement is true, and hence proving *existence* of a witness, does not imply a proof of *knowledge* of the witness. Consider, for example, the case in which common input is an integer N, and the prover wants to prove that he knows a non-trivial factor N. (Here the corresponding "statement" would be that N is composite, but this can easily be checked by the verifier offline, without the need for an interaction.)

Identification schemes are a natural application of zero knowledge. Suppose that a user wants to log in into a server. In a typical Unix setup, the user has a password

x, and the server keeps a hash $f(x)$ of the user's password. In order to log in, the user sends the password x to the server, which is insecure because an eavesdropper can learn x and later impersonate the user.

In a secure identification scheme, instead, the user generates a public-key/ secret key pair (pk, sk), the server knows only the public key pk, and the user "convinces" the server of his identity without revealing the secret key. (In SSH, for example, (pk, sk) are the public key/ secret key pair of a signature scheme, and the user signs a message containing a random session identifier in order to "convince" the server.)

If f is a one-way function, then a secure identification scheme could work as follows: the user picks a random secret key x and lets its public key be $f(x)$. To prove its identity, the user engages in a *zero knowledge proof of knowledge* with the server, in which the user plays the prover, the server plays the verifier, and the protocol establishes that the user knows an inverse of $f(x)$. Hence, the server would be convinced that only the actual person would be able to log in, and moreover from the point of view of the user he/she will not be giving away any information the server might maliciously utilize after the authentication.

This example is important to keep in mind as every feature in the definition of the protocol has something desirable in the protocol of this model.

The main application of zero knowledge proofs is in the theory of multi party protocols in which multiple parties want to compute a function that satisfies certain security and privacy property. One such example would be a protocol that allow several players to play online poker with no trusted server. By such a protocol, players exchange messages to get the local view of the game and also at the end of the game to be able to know what is the final view of the game. We would like that this protocol stays secure even in the presence of malicious players. One approach to construct such a secure protocol is to first come up with a protocol that is secure against "honest but curious" players. According to this relaxed notion of security, nobody gains extra information provided that everybody follows the protocol. Then one provides a generic transformation from security against "honest but curious" to security against malicious user. This is achieved by each user providing a ZKP at each round that in the previous round he/she followed the protocol. This would on one side convince the other players that no one is cheating and on the other side the player presenting the protocol would provide no information about his own cards. This forces apparent malicious players to act honestly, as only they can do is to analyze their own data. At at the same time this is also not a problem for the honest players.

2 The Graph Non-Isomorphism Protocol

We say that two graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ are *isomorphic* if there is a bijective relabeling $\pi : V \to V$ of the vertices such that the relabeling of G_1 is the same graph as G_2, that is, if

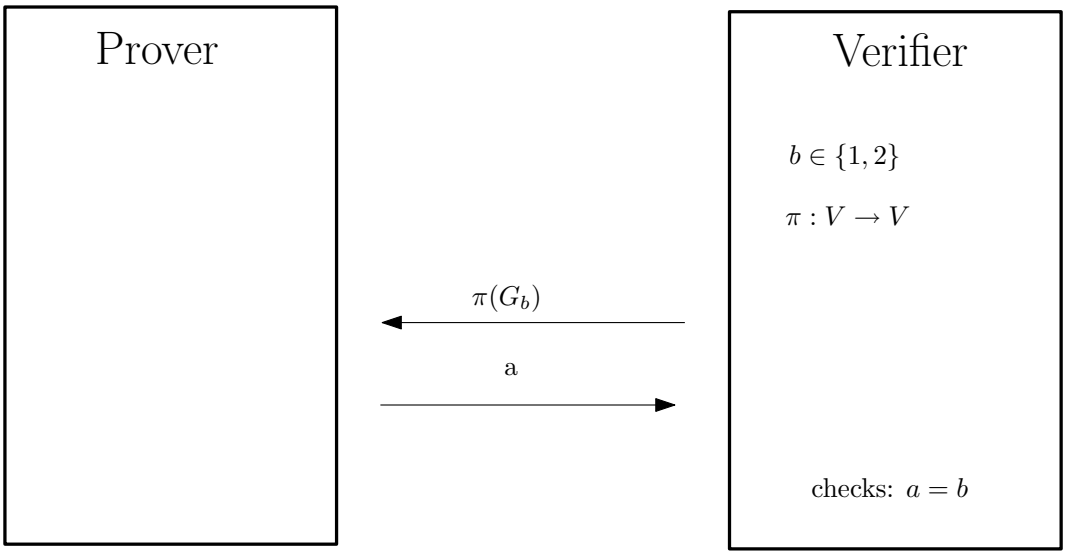
$$(u, v) \in E_1 \Leftrightarrow (\pi(u), \pi(v)) \in E_2$$

We call $\pi(G_1)$ the graph that as an edge $(\pi(u), \pi(v))$ for every edge (u, v) of E_1.

The graph isomorphism problem is, given two graphs, to check if they are isomorphic.

It is believed that this problem is not NP-complete however algorithm that would run faster than $\mathcal{O}\left(2^{\sqrt{N}}\right)$ is not known.

Here we describe an interactive protocol in which a prover can "convince" a verifier that two given graphs are not isomorphic, and in which the verifier only makes questions for which he already knows an answer, so that, intuitively, he gains no new knowledge from the interaction. (We will give a precise definition later, but we will not prove anything formal about this protocol, which is presented only for intuition.) For the prover, unfortunately, we only know how to provide an exponential time implementation. However the verifier algorithm, is very efficient.



- Common input: two graphs $G_1 = (V, E_1)$, $G_2 = (V, E_2)$; the prover wants to convince the verifier that they are *not* isomorphic
- The verifier picks a random $b \in \{1, 2\}$ and a permutation $\pi : V \to V$ and sends $G = \pi(G_b)$ to the prover

- The prover finds the bit $a \in \{1, 2\}$ such that G_a and G are isomorphic sends a to the verifier
- The verifier checks that $a = b$, and, if so, accepts

Theorem 1 *Let P be the prover algorithm and V be the verifier algorithm in the above protocol. Then*

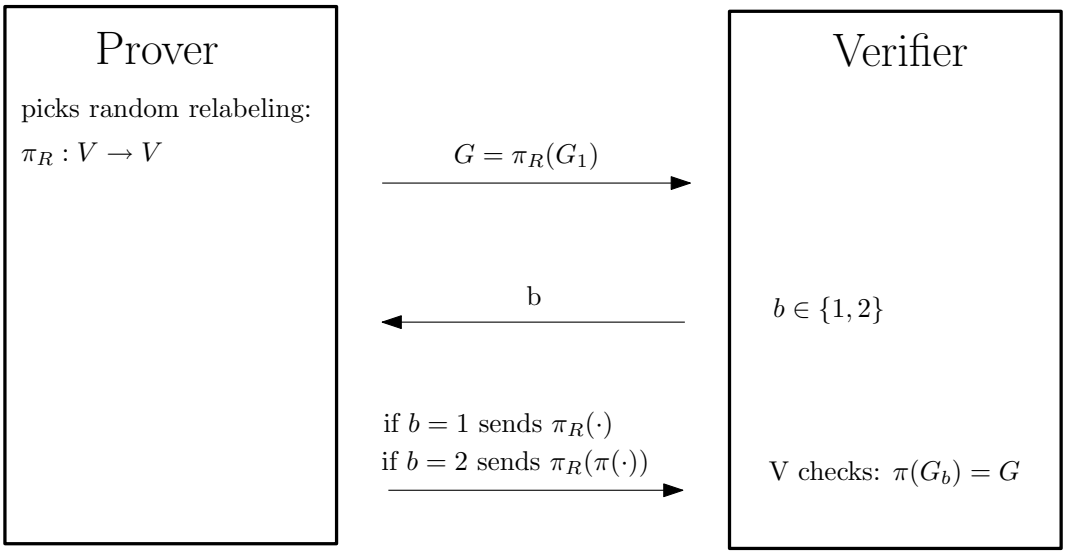
1. *If G_1, G_2 are not isomorphic, then the interaction $P(x) \leftrightarrow V(x)$ ends with the verifier accepting with probability 1*
2. *If G_1, G_2 are isomorphic, then for every alternative prover strategy P^*, of arbitrary complexity, the interaction $P^*(x) \leftrightarrow V(x)$ ends with the verifier accepting with probability $1/2$*

The first part of the theorem is true as for every permutation $\pi(G_1)$ is not isomorphic to G_2 and similarly for every permutation $\pi(G_2)$ is not isomorphic to G_2, therefore if G_1 and G_2 are not isomorphic no relabeling of G_1 can make it isomorphic to G_2. Since the prover runs in exponential time he can always find out which graph the verifier has started from and therefore the prover always gives the right answer.

The second part of the theorem is true as there exist permutation π^* such that $\pi^*(G_2) = G_1$. Then if verifier picks a random permutation π_R then the distribution we obtain by $\pi_R(\pi^*(G_2))$ and the distribution $\pi_R(G_1)$ are exactly the same as both are just random relabelling of, say, G_1. This fact is analogous to the fact that if we add a random element from the group to some other group element we get again the random element of the group. Therefore here the answer of the prover is independent on b and the prover succeeds with probability half. This probability of $\frac{1}{2}$ can be reduced to 2^{-k} by repeating the protocol k times. It is important to notice that this protocol is ZK since the verifier already knows the answer so he learns nothing at the end of interaction. The reason why the verifier is convinced is because the prover would need to do something that is information theoretically impossible if the graphs are isomorphic. Therefore, it is not the answers themselves that convince the prover but the fact that prover can give those answers without knowing the isomorphism.

3 The Graph Isomorphism Protocol

Suppose now that the prover wants to prove that two given graphs G_1, G_2 are isomorphic, and that he, in fact, knows an isomorphism. We shall present a protocol for this problem in which both the prover and the verifier are efficient.



- Verifier's input: two graphs $G_1 = (V, E_1)$, $G_2 = (V, E_2)$;
- Prover's input: G_1, G_2 and permutation π^* such that $\pi^*(G_1) = G_2$; the prover wants to convince the verifier that the graphs are isomorphic
- The prover picks a random permutation $\pi_R : V \to V$ and sends the graph $G := \pi_R(G_2)$
- The verifier picks at random $b \in \{1, 2\}$ and sends b to the prover
- The prover sends back π_R if $b = 1$, and $\pi_R(\pi^*(\cdot))$ otherwise
- The verifier checks that the permutation π received at the previous round is such that $\pi(G_b) = G$, and accepts if so

Theorem 2 *Let P be the prover algorithm and V be the verifier algorithm in the above protocol. Then*

1. *If G_1, G_2 are isomorphic, then the interaction $P(x) \leftrightarrow V(x)$ ends with the verifier accepting with probability 1*
2. *If G_1, G_2 are not isomorphic, then for every alternative prover strategy P^*, of arbitrary complexity, the interaction $P^*(x) \leftrightarrow V(x)$ ends with the verifier accepting with probability* $1/2$

The first part is clear from the construction.

What happens if G_1 and G_2 are not isomorphic and the prover is not following the protocol and is trying to cheat a verifier? Since in the first round the prover sends a

graph G, and G_1 and G_2 are not isomorphic, then G can not be isomorphic to both G_1 and G_2. So in second round with probability at least half the verifier is going to pick G_b that is not isomorphic to G. When this happens there is nothing that the prover can send in the third round to make the verifier accept, since the verifier accepts only if what prover sends in the third round is the isomorphism between G and G_b. Hence the prover will fail with probability a half at each round and if we do the same protocol for several rounds the prover will be able to cheat only with exponentially small probability.

Definition 3 *A protocol defined by two algorithms P and V is an interactive proof with efficient prover, for a decision problem if:*

- ***(Completeness)*** *for every input x for which the correct answer is YES, there is a witness w such that $P(x,w) \leftrightarrows V(x)$ interaction ends with V accepting with probability one.*
- ***(Soundness)*** *for every input x for which answer is NO, for algorithm P^* of arbitrary complexity $P^*(x,w) \leftrightarrows V(x)$ interaction ends with V rejecting with probability at least half (or at least $1 - \frac{1}{2^k}$ if protocol repeated k times)*

So the graph isomorphism protocol described above is an *interactive proof with efficient prover* for the graph isomorphism protocol.

We now formalize what we mean by the verifier *gaining zero knowledge* by participating in the protocol. The interaction is ZK if the verifier could simulate the whole interaction by himself without talking to the prover.

Definition 4 (Honest Verifier Perfect Zero Knowledge) *A protocol (P,V) is Honest Verifier Perfect Zero Knowledge with simulation complexity s for a decision problem if there is an algorithm $S(\cdot)$ that has complexity at most s, such that $\forall x$ for which the answer is YES, $S(x)$ samples the distribution of $P(x,w) \leftrightarrows V(x)$ interactions for every valid w.*

Therefore the simulator does not know the witness but it is able to replicate the interaction between the prover and the verifier. One consequence of this is, the protocol is able to simulate all possible interactions regardless of what particular witness the prover is using. Hence the protocol does the same regardless of witness. This *witness indistinguishability* property is useful on its own.

Now consider an application in which the user is the prover and the server is the verifier. For this application of ZKP it is not sufficient that honest V does not learn anything following the protocol but also that if the verifier is not honest and does not follow the protocol he will still not be able learn anything from the prover.

Therefore the full definition of zero knowledge is the following.

Definition 5 (Perfect Zero Knowledge) *A prover algorithm P is (general) Perfect Zero Knowledge with simulation overhead $so(\cdot)$ for a decision problem if*

- *for every algorithm V' of complexity at most t there is a simulator algorithm S' of complexity at most $so(t)$,*
- *such that for every x for which the answer is YES, and every valid witness w, $S'(x)$ samples $P(x,w) \leftrightarrows V'(x)$.*

(In an asymptotic setting, we would want $so(t)$ to be polynomial in t. Typically, we have $so(t) \leq O(t) + n^{O(1)}$.)

So from the prover's viewpoint the protocol is always safe, since even if the verifier does not follow the protocol he would be able to gain only the information that he (the verifier) would gain otherwise anyway, by running the simulator himself.

The zero knowledge property is purely the property of the prover algorithm since it is quantified of over all witnesses, all inputs, and all verifier algorithms. Symmetrically the soundness property was the property of the verifier algorithm since the verifier would get convinced with high probability only if the property is really true, regardless whether the prover is malicious or not.

In the next class we will establish the fact that the prover algorithm in the graph isomorphism protocol described above is (general) perfect zero knowledge.