CS 276 – Cryptography Sept 24, 2014

Lecture 7: Interactive Proofs and Zero Knowledge

Instructor: Sanjam Garg *Scribe: Manuel Sabin*

1 Interactive Proofs

So, what is a proof? We generally have an intuition that a proof can be written down and that, when read, it convinces the reader of some truth.
And, as we step into the rigor of mathematics, truth becomes an objective quality based on a set of assumptions or axioms. A proof, then, reduces a statement to those axioms or to something else that has already been shown to reduce to those axioms and we can start rigorously defining a proof.
As humans, however, we begin to give proofs more structure to accommodate our limitations. The proof that there are infinitely many primes, for example, would not be satisfying or feasible if its proof was to write out each of the infinitely many primes; not only could the prover never finish their proof, but the verifier would never be able to fully be convinced. In this case, it's become natural to have proofs be finite and generally even succinct.
"Succinct," though, has taken on a more defined concept since the advent of computers. Since computers can "read" much quicker than humans, the concept of computational complexity has been developed to gain an idea of what is feasible for a computer. A complexity theoretic concept has been defined to encapsulate the statements that are generally considered "easy" for a computer to verify their veracity when given proofs for them: the class NP.

Definition 1 (NP-Verifier) *A language L has an NP-verifier if $\exists$ a verifier $\mathcal{V}$ that is polynomial time in $|x|$ such that:*

- *Completeness: $\forall x \in L$, $\exists$ a proof π s.t. $\mathcal{V}(x, \pi) = 1$*
- *Soundness: $\forall x \notin L$ $\forall$ purported proof π we have $\mathcal{V}(x, \pi) = 0$*

That is, the conventional idea of a proof is formalized in terms of what a computer can efficiently verify. So a set of statements considered true (e.g. in a language L) is complete and sound if a proof can be written down that can be "easily" and rigorously verified if and only if a statement is in the language.
The role computers took in proofs was not only to more rigorously define what it means to be an "efficient" proof, but it also changed the focus of many proofs; rather than proving general statements, many proofs were showing that specific instances of a problem were in fact true, as in the case of NP problems.
Some problems are easily seen to be in NP. Graph Isomorphism (GI), for example, is the problem of deciding if two graphs are identical in terms of how vertices are connected by edges; that is, if two graphs are simply renamings of the the vertices of the other. No algorithm is known that can solve this general problem efficiently. However if two graphs are isomorphic, writing down how each vertex is relabled so that the two graphs are identical is a proof that the two are isomorphic that is easily verifiable.
And so, considering the language consisting of tuples of graphs that are indeed isomorphic, we have that this language has an NP-verifier.

If we instead consider the language of tuples of graphs that are *not* isomorphic, we correspond to the Graph Non-Isomorphism (GNI) problem. It is harder to think of an NP-verifier for this language; indeed, what proof could be given that two graphs are not isomorphic besides giving every possible isomorphism of the first graph to verify that none of them are the second graph? An efficient proof that can be written down, in fact, has not been discovered in the general case and this problem is not known to be in NP.

Computational thinking, however, not only unlocked the concept of "efficient" proofs for computers, but also began to challenge conventional proofs in a much bigger way. What if there really *was* a way to prove two graphs were not isomorphic efficiently?

Example. To formalize GNI, note that G_0 and G_1 are isomorphic, denoted $G_0 \cong G_1$, if $\exists$ an isomorphism $f : V(G_0) \to V(G_1)$ s.t. $(u, v) \in E(G_0)$ iff $(f(u), f(v)) \in E(G_1)$, where $V(G)$ and $E(G)$ are the vertex and edge sets of some graph G.

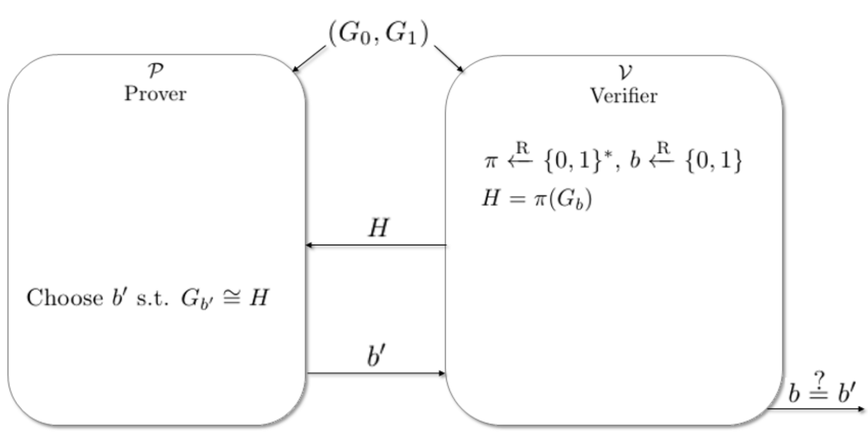
And so G_0 and G_1 are not isomorphic, $G_0 \not\cong G_1$, if $\nexists$ any such f, and $GNI = \{(G_0, G_1)|\ G_0 \not\cong G_1\}$ is the language we would like a proof system for.

Unfortunately, it's hard to imagine a prover that can write down an efficient proof to be verified and so an NP-verifier system seems inadequate. Even if the prover is all-powerful and can look at each isomorphism of each graph and see how there is no isomorphism between them, there is no known way to convey that information and convince a verifier who is not also all-powerful and who must read and be convinced of the proof "efficiently."

Instead, consider how a lying prover might be exposed if the graphs actually are isomorphic. How could $G_0 \cong G_1$ be used to stump a prover, even if they're computationally all-powerful?

Say we randomly rename the vertices of one of the graphs and give it to the prover; can they tell which relabeled graph they received? If they were not isomorphic then G_0 and G_1 would be structurally different and the prover, being all-powerful, could recognize this and determine which graph they were given. However, they *are* isomorphic and so a random relabeling of one of them could just have easily been a relabeling of the other. And the prover, all-powerful though it may be, has no way of distinguishing which graph it was given. So the prover has only a 50/50 chance of guessing which graph they were given, and repeatedly testing them with this idea forces them to be an extremely lucky guesser or to be exposed as a liar!

If they consistently answer correctly, however, it would be hard to remain skeptical against $G_0 \not\cong G_1$ as they beat the odds to almost impossible limits. And so this interaction can "prove" very strongly to the verifier that $(G_0, G_1) \in$ GNI. Consider the protocol we can define from this:



- Completeness: If $(G_0, G_1) \in$ GNI, then the all-powerful $\mathcal{P}$ can distinguish isomorphisms of G_0 against those of G_1 and can always return the correct b'. Thus, $\mathcal{V}$ will always output 1 for this case.

- Soundness: If $(G_0, G_1) \notin$ GNI, then it is equiprobable that H is a random isomporphism of G_0 as it is G_1 and so $\mathcal{P}$ must guess the correct b' and will succeed with probability $\frac{1}{2}$. Doing k rounds of this protocol means the probability of guessing the correct b' for all k rounds is $\frac{1}{2^k}$. And so the probability of $\mathcal{V}$ outputting 0 (e.g. rejecting $\mathcal{P}$'s proof at the first sign of falter) is 1-$\frac{1}{2^k} \approx 1$ for a sufficiently large k. Note that soundness is no longer perfect as it was with NP-verifiers but is instead probabilistic, with extremely high probability easily attainable.

And so the interaction between prover and verifier captures the notion of a proof system for GNI, a problem previously not known to have an efficient method of proof. By interacting, we can prove what seemed impossible to prove before!
When something this cool happens, it's generally a good idea to give it a name and some rigor.

Definition 2 (Interactive Proof System) *For a language L we have an interactive proof system if $\exists$ a pair of algorithms (or better, interacting machines) $(\mathcal{P}, \mathcal{V})$, where $\mathcal{P}$ is computationally all-powerful, $\mathcal{V}$ is polynomial in $|x|$, and both can flip coins, such that:*

- *Completeness:* $\forall x \in L$
$$\Pr_{\mathcal{P},\mathcal{V}}[Output_{\mathcal{V}}(\mathcal{P}(x) \leftrightarrow \mathcal{V}(x)) = 1] = 1$$

- *Soundness:* $\forall x \notin L$, $\forall \mathcal{P}^*$
$$\Pr_{\mathcal{V}}[Output_{\mathcal{V}}(\mathcal{P}^*(x) \leftrightarrow \mathcal{V}(x)) = 1] < neg(|x|)$$

And so as the prover (a non-deterministic Turing machine) and the verifier (a PPT Turing machine) interact, new things can be accomplished in the class of languages that have interactive proof systems: IP. As exciting as this insight is, is there even more that can be done from this concept of interaction?

2 Zero Knowledge Proofs

Unfortunately (fortunately?), there aren't real-life instances of all-powerful provers that we know of. And for cryptography we must make more reasonable assumptions about the provers. In this case we will assume provers are also bounded to be "efficient."
Previously, if a prover wanted to prove that two graphs, G_0 and G_1 were isomorphic, it would use its all-powerfulness to find the isomorphic mapping between the two graphs and give it to the verifier to complete the proof. But now, being computationally bounded, the prover is in the same boat as the verifier and can find a proof no better than the verifier can. In order for the prover to be able to prove something that the verifier cannot find out on their own, the prover must have some extra information. If, for example, the prover simply knew the isomorphism between the graphs, this would be the sufficient extra information it needs to enact the proof. That's a rather boring proof though. We have interaction now! Can't we do something fancier?

What if the prover wanted to prove that two graphs were isomorphic but didn't want to fully reveal the isomorphism that they know. If they're lying and don't know an isomorphism is their a way we can exploit them again?

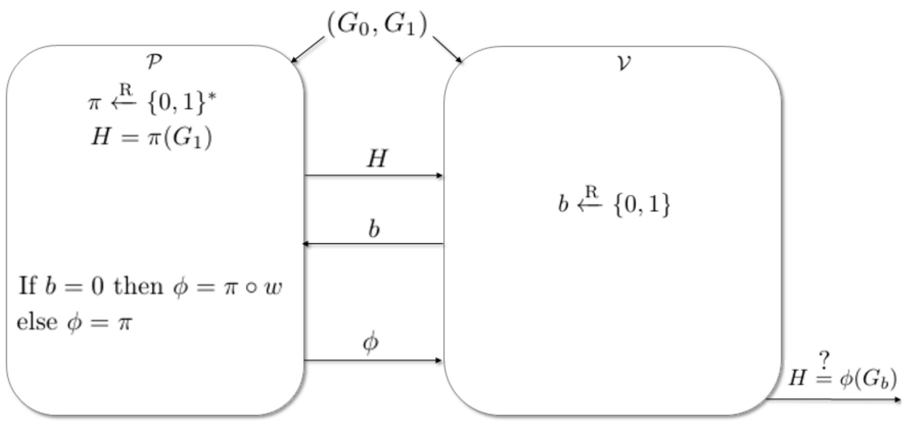
When G_0 and G_1 are isomorphic, the isomorphism between them would be a *witness*, w, to that fact, that can be used in the proof. Unfortunately, the prover is being stubborn and won't just tell us that isomorphism, $w : V(G_0) \to V(G_1)$, that they claim to have. The prover is comfortable however giving us a "scrambled" version, ϕ, of w as long as it doesn't leak any information about their precious w. For example, the prover is willing to divulge $\phi = \pi \circ w$ where π is a privately chosen random permutation of $|V| = |V(G_0)| = |V(G_1)|$ vertices. Since π renames vertices completely randomly, it scrambles what w is doing entirely and ϕ is just a random permutation of $|V|$ elements. At this point, we might be a little annoyed at the prover since we could have just created a random permutation on our own. This might give us an idea on how to gain a little more information however, even though we gained none here:

If we want to be convinced that ϕ really is of the form $\pi \circ w$, thus containing w in its definition, and isn't just a completely random permuation, we can note that if it is of that form then $\phi(G_0) = \pi(w(G_0)) = \pi(G_1)$ (since w being an isomorphism implies that $w(G_0) = G_1$). Note that we started with a mapping on input G_0 and ended with a mapping on input G_1. With an isormphism, one could get from one graph to the other seamlessly; if the prover *really* has the isomorphism it claims to have, then it should have no problem displaying this ability. So, what if we force the prover to give us $H = \pi(G_1)$ just after randomly choosing its π and then let it show us its ability to go from G_1 to G_0 with ease: give us a ϕ so that $\phi(G_0) = \pi(G_1) = H$. The only way the prover can give a mapping that jumps from G_0 to G_1 in such a way is if they know an isomorphism; if the prover could find a ϕ efficiently but *didn't* know an isomorphism then they would have be able to see that $\pi^{-1}(\phi(G_0)) = G_1$ and thus have $\pi^{-1} \circ \phi$ as an isomorphism from G_0 to G_1, which would contradict the assumed hardness of finding isomorphisms in the GI problem. So by forcing the prover to give us H as we've defined and to produce a ϕ so that $\phi(G_0) = H$, we've found a way to expose provers that don't really have an isomorphism and we can then be convinced that they really do know w when they pass our test. And the prover didn't directly tell us w, so they may be able to salvage some secrecy!

But not everything is airtight about this interaction. Why, for instance, would the prover be willing to provide $H = \pi(G_1)$ when they're trying to divulge as little information as possible? The prover was comfortable giving us ϕ since we could have just simulated the process of getting a completely random permutation of vertices ourselves, but couldn't the additional information of H reveal information about w? At this point, the annoyed feeling may return as we realize that, $H = \pi(G_1) = \pi'(G_0)$, for some π', is just a random isomorphic copy of G_0 *and* G_1 as long as $G_0 \cong G_1$; we could have just chosen a random π', set $H = \pi'(G_0)$, and let $\phi = \pi'$ and would have created our very own random isomorphic copy, H, of G_1 that satisfies our test condition $H = \phi(G_0)$ just like what we got from our interaction with the prover. We couldn't have gained any new information from the prover because we could have run the whole test on our own!

Well, something must be wrong; we couldn't have been convinced of something without gaining *any* new information. Indeed, the test has a hole in it: how can we force the prover to give us $H = \pi(G_1)$ like we asked? If the prover is lying and it knows our test condition is to verify that $H = \phi(G_0)$, the prover might just cheat and give us $H = \pi(G_0)$ so it doesn't have to use knowledge of w to switch from G_1 to G_0. And, in fact, by doing this and sending $\phi = \pi$, the prover would fool us!

To keep the prover on their toes, though, we can randomly switch whether or not we want H to equal $\phi(G_0)$ or $\phi(G_1)$. If, in our interaction, the prover must first provide their $H = \pi(G_1)$ before we let them know which we want, they then lock themselves into a commitment to either G_0 or G_1 depending on whether they're trying to cheat or not, respectively. They only have a 50% chance of committing to the same case we want on a given round and so, if they don't have w to deftly switch between G_0 and G_1 to always answer correctly, they again have to be an extremely lucky guesser if they're trying to lie.
Again, we've created an interactive scheme that can catch dishonest provers with probability 1-$\frac{1}{2^k}$ and where we always believe honest provers!



- Completeness: If $(G_0, G_1) \in$ GI and $\mathcal{P}$ knows w, then whether $\mathcal{V}$ chooses $b = 0$ or 1, $\mathcal{P}$ can always give the correct ϕ which, by definition, will always result in $H = \phi(G_b)$ and so $\mathcal{V}$ will always output 1.

- Soundness: If $(G_0, G_1) \notin$ GI, then $\mathcal{P}$ can only cheat, as discussed earlier, if the original H it commits to ends up being $\pi(G_b)$ for the b that is randomly chosen at the next step. Since b isn't even chosen yet, this can only happen by chance with probability $\frac{1}{2}$. And so the probability $\mathcal{V}$ outputs 0 is 1-$\frac{1}{2^k}$ for k rounds.

And so, again, we've correctly captured the idea of a proof by having this interaction. But there's a strange feeling that may be lingering around us...
As a verifier, we've seen some things in interacting with the prover. Surely, clever folks like ourselves must be able to glean *some* information about w after seeing enough to thoroughly convince us that the prover knows w. We've first seen H, and we've also seen the random b that we chose, along with ϕ at the end; this is our whole view of information during the interaction. But we're more bewildered than annoyed this time when we realize we could have always just chosen b and ϕ randomly and set $H = \phi(G_b)$ on our own. Again, everything checks out when $G_0 \cong G_1$ and we could have produced everything that we saw during the interaction before it even began. That is, the distribution of the random variable triple (H, b, ϕ) is identical whether it is what we saw from the prover during the interaction or it is yielded from the solitary process we just described. We've just constructed a complete interactive proof system that entirely convinces us of the prover's knowledge of w, yet we could have simulated the whole experience on our own! We couldn't have gain any knowledge about w since we didn't see anything we couldn't have manufactured on own,

yet we are entirely convinced that $(G_0, G_1) \in \text{GI}$ and that $\mathcal{P}$ knows w! And so the prover has proven something to us yet has given us absolutely zero additional knowledge!

This may feel very surprising or as if you've been swindled by a fast talker, and it very much should feel this way; it was certainly an amazing research discovery! But this is true, and it can be made rigorous:

We should first be sure what we want out of this new proof system. We of course want it to be complete and sound so that we accept proofs iff they're true. But we also want the verifier to gain zero knowledge from the interaction; that is, the verifier should have been able to simulate the whole experience on its own without the verifier. Finally, we would also like all witnesses to a true statement to each be sufficient to prove the veracity of that statement and so we let R be the relation s.t. $x \in L$ iff $\exists$ a witness w s.t. $(x, w) \in R$. We can then gather all witness by defining $R(x)$ to be the set of all such witnesses.

Definition 3 (Honest Verifier Zero Knowledge Proof [HVZK]) *For a language L we have a (perfect) HVZK proof system w.r.t. witness relation R if $\exists$ an interactive proof system, $(\mathcal{P}, \mathcal{V})$ s.t. $\exists$ a PPT machine $\mathcal{S}$ (called the simulator) s.t. $\forall x \in L$, $\forall w \in R(x)$ the following distributions are identical:*

$$View_{\mathcal{V}}(\mathcal{P}(x, w) \leftarrow \mathcal{V}(x))$$

$$\mathcal{S}(x)$$

where $View_{\mathcal{V}}(\mathcal{P}(x, w) \leftarrow \mathcal{V}(x))$ is the random coins of $\mathcal{V}$ and all the messages $\mathcal{V}$ saw.

There's an interesting progression of the requirements of a proof system: Completeness, Soundness, and the Zero Knowledge property. Completeness first cares that a prover-verifier pair exist and can capture all true things as a team that works together; they both honestly obey the protocol trying prove true statements. Soundness, however, assumes that the prover is a liar and cares about having a strong enough verifier that can stand up to any type of prover and not be misled. Finally, Zero Knowledge assumes that the verifier is hoping to glean information from the proof to learn the prover's secrets and this requirement makes sure the prover is clever enough that it gives no information away in its proof.

Unlike the soundness' requirment for a verifier to combat *all* malicious provers, HVZK is only concerned with the verifier in the original prover-verifier pair that follows the set protocol. Verifiers that stray from the protocol or cheat, however, are captured in the natural generalization to Zero Knowledge proofs. These are mostly discussed (including auxiliary inputs) in the next class, although the first definition is given below:

Definition 4 (Zero Knowledge Proof [ZK]) *For a language L we have a (perfect) ZK proof system w.r.t. witness relation R if $\exists$ an interactive proof system, $(\mathcal{P}, \mathcal{V})$ s.t. $\exists$ a PPT machine $\mathcal{S}$ (called the simulator) s.t. $\forall x \in L$, $\forall w \in R(x)$, $\forall \mathcal{V}^*$, the following distributions are identical:*

$$View_{\mathcal{V}^*}(\mathcal{P}(x, w) \leftarrow \mathcal{V}^*(x))$$

$$\mathcal{S}^{\mathcal{V}^*}(x)$$

where $\mathcal{S}^{\mathcal{V}^*}(x)$ is the simulator with oracle access to $\mathcal{V}^*$.