1 Recap: Interactive Proofs

Definition 1.1. An *interactive proof system* with *soundness error* $s \in [0,1]$ for a language $L \subseteq \{0,1\}^*$ is a pair of algorithms: a (possibly computationally unbounded) prover P, and a ppt verifier V, having the following properties:

1. *Completeness* ("the specified prover convinces the specified verifier of a true statement"): for all $x \in L$, $\text{out}_V[P(x) \leftrightarrow V(x)] = 1$, with probability 1.
2. *Soundness*: ("*no* possibly cheating prover can convince the specified verifier of a false statement"):

$$\Pr\left[\text{out}_V[P^*(x) \leftrightarrow V(x)] = 1\right] \leq s.$$

Last time we studied an interactive proof for the graph nonisomorphism problem (GNI). Remember the definition: $\mathsf{GNI} = \{(G_0, G_1) : G_0 \not\equiv G_1\}$, where $G_0 \equiv G_1$ if there exists a bijection $\pi : V_0 \to V_1$ such that $(v_0, v_1) \in E_0$ if an only if $(\pi(v_0), \pi(v_1)) \in E_1$. The protocol is as follows:

$P(G_0, G_1)$		$V(G_0, G_1)$
		$b \leftarrow \{0,1\}$
		choose random permutation π on G_b's vertices
	$\xleftarrow{H = \pi(G_b)}$	
find b' such that $H \equiv G_{b'}$		
	$\xrightarrow{b'}$	
		accept iff $b = b'$

We already proved that this protocol is complete (if the graph are not isomorphic, then the unbounded prover always convinces the verifier) and that the protocol is sound (if the graphs are isomorphic, then the best any prover can do is to convince the verifier with one-half probability).

2 Zero-Knowledge Proofs

Notice a curious property of the above protocol: the prover just replies with $b' = b$, the verifier's own random bit. Intuitively, therefore, V doesn't seem to get anything from its interaction with P that it "doesn't already know itself" — aside from the fact that the theorem is true (because the prover gives a convincing proof). Properly formalized, this property is called "zero knowledge."

2.1 Honest-Verifier Zero-Knowledge

Definition 2.1. An interactive proof system (P, V) for a language L is said to be *honest-verifier zero-knowledge* if there exists an nuppt simulator $\mathcal{S}$ such that for all $x \in L$,

$$\text{view}_V[P(x) \leftrightarrow V(x)] \overset{s}{\approx} \mathcal{S}(x),$$

where view_V is the "view" of the verifier in the interaction, consisting of its input x, its random coins r_V, and the sequence of the prover's messages.

The idea behind the definition is that given an $x \in L$, it is possible to simulate everything the (honest) verifier "sees" in the interaction *without the help of the unbounded prover*. Said another way, anything revealed by the prover in the interaction could have been generated by a simulator that knows nothing more than the verifier itself.

Let us take a closer look at the definition of view_V. The verifier's "view" is the tuple of all its various inputs throughout the execution of the protocol. These are precisely its input x, its random coins, and the prover's messages. Observe that given these values, the messages that V sends to P can be generated deterministically, so we do not need to include them in the view.

Lemma 2.2. *The above protocol for the* GNI *problem is honest-verifier zero-knowledge.*

Proof. To prove this lemma we define a ppt simulator $\mathcal{S}$ that is able to generate a transcript that is indistinguishable from (actually, identical to) the view of V. The simulator $\mathcal{S}(G_0, G_1)$ works as follows: first, choose $b \leftarrow \{0,1\}$ and a random permutation π on G_b's vertices. Then, output the view $(x = (G_0, G_1), r_V = (b, \pi), b' = b)$.

We show that when $(G_0, G_1) \in \mathsf{GNI}$, we have that $S(G_0, G_1) \equiv \text{view}_V[P(x) \leftrightarrow V(x)]$. The random coins (b, π) are distributed identically to V's random coins. And since we are assuming that $(G_0, G_1) \in \mathsf{GNI}$, the prover always answers with the message $b' = b$. This completes the proof. □

2.2 Full Zero-Knowledge

The above definition says that the verifier gains no knowledge from the interaction, as long as it runs the prescribed algorithm V. But what if the verifier tries to gain some knowledge from its interaction with the prover by *deviating* from the prescribed protocol? We should consider an arbitrary (but efficient) *cheating* verifier V^*, and show that its view can be efficiently simulated as well.

Definition 2.3. An interactive proof system (P, V) for a language L is *zero-knowledge* if for every nuppt V^* there exists an nuppt simulator $\mathcal{S}$ such that for all $x \in L$,

$$\text{view}_{V^*}[P(x) \leftrightarrow V^*(x)] \overset{s}{\approx} \mathcal{S}(x).$$

Does our protocol for GNI enjoy full zero-knowledge? Let's try to create a simulator $\mathcal{S}$. Since V^* is arbitrary, we do not know how V^* will construct its first message (the graph H), so the only thing $\mathcal{S}$ seems to be able to do is to choose random coins r_{V^*} and run V^* using those random coins to get a message H. This H can be anything (it might not even be a graph at all!). If it's a malformed message, then $\mathcal{S}$ can do exactly what the honest prover P would do: abort. But what if H is a valid message? Note that $\mathcal{S}$ can't hope to determine the correct graph just by "inspecting" the random coins r_{V^*}, because we have no idea how V^* generates H from its random coins. So how can $\mathcal{S}$ know if H is isomorphic to G_0 or G_1, without solving the graph isomorphism problem?

The problem here is that the protocol in fact *may not be* zero-knowledge. Notice that in this protocol, the prover is acting as an *isomorphism oracle*, and V^* might be able to exploit the prover to get some extra knowledge that it didn't have before. Suppose for example that V^* has H hardcoded into it, and wants to know if $H \equiv G_0$ or $H \equiv G_1$ (or neither). Then V^* can simply send H to the prover, and from the answer V^* will learn that $H \equiv G_{b'}$, or that H is isomophic to neither of G_0, G_1 (if the prover happens to abort).

3 Graph Isomorphism

It is possible to give a zero-knowledge proof for graph nonisomorphism, but the protocol and proof are quite involved (the solution effectively forces V^* itself to prove that H is indeed isomorphic to one of G_0 or G_1, which effectively eliminates the problem we identified above). Instead, let's look at a different problem for which we will be able to give a simple zero-knowledge proof. We define the *graph isomorphism* problem $\mathsf{GI} = \{(G_0, G_1) : G_0 \equiv G_1\}$. This problem is just the complement of the GNI problem. Can we create a zero-knowledge interactive proof system for this problem? Before just giving the solution, let's develop some intuition for how it might work.

Support that a spelunker wants to prove that a certain cave contains a "loop," but without giving any other knowledge about the structure of the cave. A solution might be the following: the verifier stays at any point of the claimed loop, and the prover moves to some hidden part of the loop. When the prover announces that he is ready, the verifier will choose one of the two directions and will challenge the prover to come to him from that direction. If the prover knows the loop, he will be able to find a way to the verifier, while if there is no loop the prover will be able to walk to the verifier from only one direction and will succeed only half of the time.

We will use this basic idea for the GI problem. The prover will generate a random graph H isomorphic to the two graphs, and the verifier will challenge it to give a permutation proving that H is isomorphic to G_b, for b chosen at random. Let $\text{iso}(G)$ denote the set of all graphs isomorphic to a graph G.

$P(G_0, G_1)$		$V(G_0, G_1)$
Choose random $H \leftarrow \text{iso}(G_0) = \text{iso}(G_1)$		
	$\xrightarrow{H}$	
		Choose $b \leftarrow \{0, 1\}$
	$\xleftarrow{b}$	
Choose *random* perm. ρ s.t. $H = \rho(G_b)$		
	$\xrightarrow{\rho}$	
		Accept iff $H = \rho(G_b)$

Implicitly, V should check that ρ really is a legal permutation on G_b's vertices, and reject if it is not; is important for soundness. Note also that it is very important that when replying to the challenge bit b, the honest prover P chooses a *random* permutation ρ such that $H = \rho(G_b)$, and not just (for example) the first one it encounters (since there may be more than one). This will turn out to be crucial for the proof of zero-knowledge (Claim 3.2 below).

Theorem 3.1. *The above protocol for the* GI *problem is a zero-knowledge interactive proof system.*

Proof. First we show completeness: if $G_0 \equiv G_1$, then $G_0 \equiv H \equiv G_1$ and P can always find an appropriate ρ such that V accepts.

Next we show soundness, and calculate the probability that V accepts when $G_0 \not\equiv G_1$ and interacting with some arbitrary P^*. For any H output by P^*, we have that $H \equiv G_{b'}$ for at most one b'. The verifier chooses $b \in \{0, 1\}$ at random, and if $b \neq b'$ then there is *no* reply which will make V accept. We can conclude that the soundness error is at most $\frac{1}{2}$.

Now we prove that the protocol is zero-knowledge. To do this, let V^* be an arbitrary nuppt algorithm. Unlike in our above proof of *honest-verifier* zero-knowledge for the GNI problem, we do not know how V^* will choose its challenge bit b — it might introduce some bias, or choose b to depend on the prover's initial

message H in some bizarre way. Our simulator $\mathcal{S}$ will work by preparing a message H so that it knows how to answer *just one* of V^*'s possible challenges. If V^* happens to make that challenge, the simulator answers and outputs the view; otherwise, it tries again from scratch by re-running ("rewinding") the verifier with fresh new choices until it succeeds.

More formally, $\mathcal{S}(G_0, G_1)$ works as follows:

repeat
  Choose a random permutation ρ and $b \leftarrow \{0, 1\}$
  Let $H = \rho(G_b)$
  Run $V^*(G_0, G_1)$ with fresh random coins r_{V^*}, and send H as the prover's first message to V^*
  Receive a bit b^* from V^* (or if V^* sends a malformed message, just output the view so far)
until $b^* = b$, or we've looped more than n times
return $(x = (G_0, G_1), r_{V^*}, H, \rho)$

We now need to show that for any $x = (G_0, G_1) \in \mathsf{GI}$,

$$\text{view}_{V^*}[P(x) \leftrightarrow V^*(x)] \stackrel{s}{\approx} \mathcal{S}(x).$$

This follows by Claims 3.2 and 3.3 below, because conditioned on $\mathcal{S}$ succeeding in some iteration, its output is identically distributed with the view of V^*, and it fails in all n iterations with probability at most 2^{-n}. □

Claim 3.2. *For any $x = (G_0, G_1) \in \mathsf{GI}$, conditioned on $\mathcal{S}(x)$ succeeding in a particular iteration, its output is identically distributed with view$_{V^*}[P(x) \leftrightarrow V^*(x)]$.*

Claim 3.3. *For any $x = (G_0, G_1) \in \mathsf{GI}$, the probability that $\mathcal{S}(x)$ succeeds in any particular iteration is at least $1/2$.*

Proof of Claim 3.2. The output of $\mathcal{S}$ is $((G_0, G_1), r_{V^*}, H, \rho)$. By construction, r_{V^*} is uniform. Since we choose ρ uniformly at random, we get that H is uniform in $iso(G_0) = iso(G_1)$. Furthermore, conditioned on H and bit b, ρ is a *uniformly random* permutation such that $H = \rho(G_b)$. This is exactly the prover's distribution. (This is where we have used the fact that the prover returns a random ρ such that $H = \rho(G_b)$.) □

Proof of Claim 3.3. Because $G_0 \equiv G_1$, the graph $H = \rho(G_b)$ constructed by the simulator is uniform in $\text{iso}(G_0) = \text{iso}(G_1)$, and simulator $\mathcal{S}$'s bit b is statistically independent from H. Thus, for any challenge bit b^* output by V^*, we have that $\Pr[b^* = b] = 1/2$. □

4 Enhancements

A few observations on these results:

1. *Efficient provers.* The definition of an interactive proof allows the prover to be unbounded, but how could we hope to *use* such a prover in practice? To be applicable in the real world, we would like the prover to be *efficient*. Of course, if the prover is given just the same inputs as the verifier, then it can't accomplish anything that the verifier can't accomplish itself. But in many proof systems, the prover can be made efficient by giving it an *extra* input to help it do its job. (This input might represent some extra knowledge that the prover may generated in connection with the theorem, and would like to keep secret.) For example, in the GI proof we can give the prover an isomorphism σ such that $G_0 = \sigma(G_1)$, and the prover can choose $H = \pi(G_0)$ for a uniformly random permutation π. Then if V asks for an

isomorphism between H and G_0, the prover answers π; if V asks for an isomorphism between H and G_1, the prover answers $(\pi \circ \sigma)$.

An application of zero-knowledge is to identification: a prover who knows an isomorphism between the graphs (because he constructed them, for example) can give a convincing proof. Due to the zero-knowledge property, after seeing some proofs, nobody can impersonate the prover with an advantage greater than if they hadn't seen any proofs at all. (To argue this more formally require the notion of a *proof of knowledge*, which intuitively says that not only is the theorem true, but that any (possibly cheating) prover capable of giving a convincing proof must "know a reason why" (i.e., a witness). We will investigate proofs of knowledge in more detail later.)

2. *Improved soundness error.* Our soundness error is still $1/2$, which is quite bad. One obvious way to improve the error is to repeat N *independent* iterations of the proof, and accept if every iteration is acceptable. It is not too hard to show that the soundness error then becomes 2^{-N}. Furthermore, it can be shown (though it is not trivial!) that the zero-knowledge property is also preserved under such sequential repetition. A downside is that this approach is not very efficient in the number of rounds.

   Another approach is to run N copies of the proof in *parallel*, and accept if all iterations are acceptable. This is clearly more round-efficient, and also reduces the soundness error to 2^{-N} just as with sequential repetition. However, this parallel protocol scheme may *not* preserve the zero-knowledge property! To see why, notice that when we proved that the GI protocol is zero-knowledge, we created a simulator that had to "predict" the challenge bit chosen by the cheating verifier V^*. To create a simulator for this parallel protocol, it seems that we would need to guess all N challenge bits at once — otherwise, the simulator would not be able to output a full accepting transcript for the entire protocol. (Recall that V^* can choose its N challenge bits in any arbitrary way, perhaps depending on all N of the prover's initial messages.) In the sequential approach, a simulator can guess each bit one by one (without having to "rewind" all the way to the start of the protocol), but in the parallel case all the bits need to be guessed in one step, which would succeed with only 2^{-N} probability. This would translate to a simulation running time that is exponential in N.

   Note that nothing in the above argument rules out a more clever simulation strategy, so we are *not* saying that the parallel protocol is *definitely not* zero-knowledge. All we are saying is that its status is unknown, and a valid proof of the zero-knowledge property seems hard to obtain.