Notes on Zero Knowledge

1 Interactive Proofs

We define interactive proofs as in Sipser's book, Section 10.4, except that we consider probabilistic provers. (That is, in our definition, a prover takes in input an input string, a partial message history and a random input, and gives in output a next message.)

Definition 1 (IP) *For a function $k : \mathbb{N} \to \mathbb{N}$, we define $\mathbf{IP}(k(n))$ as the class of languages L such that there is a polynomial time verifier V_L and a prover P_L such that for every input string x of length n,*

- *The interaction between V_L and P_L involves at most $k(n)$ messages.*
- *If $x \in L$ then $\mathbf{Pr}[V_L \leftrightarrow P_L \text{ accepts } x] \geq 2/3$. (Completeness.)*
- *If $x \notin L$ then for every prover P we have $\mathbf{Pr}[V_L \leftrightarrow P \text{ accepts } x] \leq 1/3$. (Soundness.)*

If the probability in the completeness case is 1 (instead of 2/3), then we say that the proof system has perfect completeness, *and we denote by $\mathbf{IP}_1(k(n))$ the class of languages having proof systems with perfect completeness and $k(n)$ rounds.*

We state the following two results without proof.

Theorem 1 *If $k(n) \geq 2$, then $\mathbf{IP}(k(n)) = \mathbf{IP}_1(k(n))$.*

Theorem 2 *For every function $k()$ such that $k(n) \geq 2$ for every n, $\mathbf{IP}(2k(n)) = \mathbf{IP}(k(n))$. In particular, for every constant k, $\mathbf{IP}(k) = \mathbf{IP}(2)$.*

There are reasons to believe that $\mathbf{IP}(2)$ is equal to **NP**, and it is considered very unlikely that $\mathbf{IP}(2)$ could contain co**NP**-hard problems.

The following results are proved in Sipser's book. Let GI be the graph isomorphism problem and GNI be the graph non-isomorphism problem.

Theorem 3 *GNI $\in \mathbf{IP}(2)$.*

This gives a very strong evidence that GI is not **NP**-complete. (Otherwise GNI would be co**NP**-complete and we would have a co**NP**-complete problem in $\mathbf{IP}(2)$.) There is no known polynomial time algorithm for GI and, in fact, GI is conjectured to not be in **P**. This means that, most likely, GI neither is in **P** nor is **NP**-complete. This interesting because almost all the natural problems in **NP** are known either to be solvable in polynomial time or to be **NP**-complete.

Theorem 4 $\mathbf{IP}(n^{O(1)}) = \mathbf{PSPACE}$.

In particular, all co**NP**-complete problems have interactive proof systems. This is interesting because it would seem impossible to prove co**NP**-complete statements (that involve exponentially many special cases) using only a polynomially long interaction.

2 Zero Knowledge

Definition 2 (Honest Verifier Zero Knowledge) *A honest verifier Perfect Zero Knowledge proof system for a language L is an interactive proof (V_L, P_L) for L, as defined in the previous section, such that there is a probabilistic algorithm S (for Simulator) that runs in average polynomial time and such that for every string $x \in L$ the distribution of outputs of $S(x)$ is identical to the distribution of views of V_L of the interaction between P_L and V_L on input x.*

The class of languages that admit a honest verifier perfect zero knowledge proof system is denoted by **HVPZK**.

A *view* of V_L is described by the random input of V_L and the sequence of messages exchanged between V_L and P_L.

The definition captures the intuition that, if a protocol is HVPZK, then the verifier V_L gains no useful information from the interaction with P_L. In fact, anything that V_L might try to compute about x after interacting with P_L and receiving a proof that $x \in$L, might also be computed without interacting with P_L and using outputs of $S(x)$ instead.

The reader can verify that the interactive proof for GNI in Siper's book demonstrates that GNI is in HVPZK.

The following definition is more general and more useful in cryptographic applications.

Definition 3 (General Zero Knowledge) *A Perfect Zero Knowledge proof system for a language L is an interactive proof (V_L, P_L) for L, as defined in the previous section, such that for every polynomial time verifier V' there is a probabilistic algorithm S' (for Simulator) that runs in average polynomial time and such that for every string $x \in L$ the distribution of outputs of $S'(x)$ is identical to the distribution of views of V' of the interaction between P_L and V' on input x.*

The class of languages that admit a perfect zero knowledge proof system is denoted by **PZK**.

This stronger condition implies that, if the prover does not even trust the verifier to follow the protocol of the proof system, the prover can still deliver a convincing proof that $x \in L$ without giving away any information about x.

There are some extra details that we are not considering here but that are important. For example, it is important for cryptographic applications that the "error" probability in the completeness and soundness case be very small functions of n (typically $1/2^n$) rather than the constant 1/3. One can reduce the probability of error by repeating the protocol several times.[1] If the protocol is repeated several times, however, it is not clear that the general zero knowledge property is preserved. There is, however, a more complicated definition of Zero Knowledge that is preserved by sequential repetition. We will not get into any of these finer points.

Clearly, every problem in **BPP** is also in **PZK**, using a proof system where no message is exchanged. (Which is easy to simulate!) A few problems not believed to be in **BPP** are also in **PZK**. In particular:

Theorem 5 *$GI \in$ **PZK**.*

PROOF: Consider the following proof system. Given two graphs $G_1 = (V, E_1), G_2 = (V, E_2)$,

[1] This is similar to the problem of reducing error in **BPP** algorithms, but the analysis for proof systems is considerably more complicated.

1. The prover picks at random a permutation π and sends to the prover the graph $G = (V, E)$ where $E = \pi(E_1))$.[2]

2. The verifier picks a bit $b \in \{0, 1\}$ at random and sends it to the prover.

3. The prover replies with a permutation π' such that $E = \pi'(E_b)$. If $b = 1$ then $\pi' = \pi$, and otherwise π' is a composition of π with the permutation that shows the isomorphism between G_1 and G_2.

   The verifier accepts if the permutation π' satisfies the required property.

The protocol clearly has perfect completeness. To analyze soundness, suppose that G_1 and G_2 are not isomorphic, and let P' be a cheating prover interacting with the honest verifier. Then P' sends some graph G at the first round, and this graph cannot be isomorphic to both G_1 and G_2. At the next round, there is a probability at least $1/2$ that the verifier will choose a b such that G and G_b are not isomorphic, and then the verifier will reject because the prover will fail to show the required permutation. Thus, for every prover, the verifier accepts with probability at most $1/2$. If the verifier repeats the protocol twice, and accepts only if both repetitions are correct, then it is easy to see that the protocol has still perfect completeness and the error in the soundness condition is only $1/4$.

For the zero knowledge property, let V' be an arbitrary cheating verifier for the protocol, and consider the following simulator. On input $G_1 = (V, E_1), G_2 = (V, E_2)$,

- Pick the random input r' for verifier V', pick $r \in \{0, 1\}$, pick a random permutation $\pi : V \to V$, define $G = (V, \pi(E_r))$;
- Write "verifier has random input r'", "prover sends G to verifier";
- Simulate verifier V' given G_1, G_2 as input strings, r' as random input, and G as first message, let b be the verifier's second message;
- If $b == r$ then write "verifier sends b to the prover", "prover sends π to verifier";
- Else FAIL

One can see that, conditioned on the event that the simulation does not fail, the output of the simulator is identical to the distribution of interactions between V' and the prover. The simulator runs in polynomial time and fails with probability $1/2$. If we keep running the simulator until it does not fail, then the average running time is still polynomial, because on average we run the simulator twice. If we want to simulate two sequential runs of the protocol, then we have probability $1/4$ of not failing, and we can still repeat the simulation until it does not fail, resulting in a polynomial time simulation. $\square$

Because of the following result (that we give without proof), **NP**-complete problems are not believed to have zero knowledge proofs.

Theorem 6 $\mathbf{PZK} \subseteq \mathbf{IP}(2) \cap co\mathbf{IP}(2)$.

There is, however, a more relaxed definition of Zero Knowledge (called *Computational* Zero Knowledge) proof system that can be realized for **NP**-complete problems.

[2]Recall that if $G = (V, E)$ is a graph and $\pi : V \to V$ is a permutation, then we denote by $\pi(E)$ the set of edges of the form $(\pi(u), \pi(v))$ such that $(u, v) \in E$.