U.C. Berkeley — CS172: Automata, Computability and Complexity   Handout 6
Professor Luca Trevisan   5/2/2007

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 $\mathbf{Pr}[V_L \leftrightarrow P_L \text{ accepts } x] = 1$. (Completeness.)*
- *If $x \notin L$ then for every prover P we have $\mathbf{Pr}[V_L \leftrightarrow P \text{ accepts } x] \leq 1/2$. (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 2 *If $k(n) \geq 2$, then $\mathbf{IP}(k(n)) = \mathbf{IP}_1(k(n))$.*

Theorem 3 *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 $\mathbf{NP}$, and it is considered very unlikely that $\mathbf{IP}(2)$ could contain co$\mathbf{NP}$-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 4 *GNI $\in \mathbf{IP}(2)$.*

This gives a very strong evidence that GI is not $\mathbf{NP}$-complete. (Otherwise GNI would be co$\mathbf{NP}$-complete and we would have a co$\mathbf{NP}$-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 $\mathbf{P}$. This means that, most likely, GI neither is in $\mathbf{P}$ nor is $\mathbf{NP}$-complete. This interesting because almost all the natural problems in $\mathbf{NP}$ are known either to be solvable in polynomial time or to be $\mathbf{NP}$-complete.

Theorem 5 $\mathbf{IP}(n^{O(1)}) = \mathbf{PSPACE}$.

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

2 Zero Knowledge

Definition 6 (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 7 (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 $1/2$. One can reduce the probability of error by repeating the protocol several times. 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.

There a few problems that are in **PZK** and that are not believed to admit polynomial time algorithms. In particular:

Theorem 8 *$GI \in$* **PZK**.

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

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

[1]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 $(u, v) \in E$.

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 9 **PZK** $\subseteq$ **IP**$(2) \cap co$**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.

3 Proofs of Knowledge

Let $R(\cdot,\cdot)$ be an NP relation, that is, a relation that is computable in polynomial time and such that there is a polynomial p such that if $R(x,y)$ then $|y| \leq p(|x|)$. The language L associated to an NP relation R is the language $\{x : \exists y. R(x,y)\}$. For example, consider the graph isomorphism relation $GIR((G_1,G_2),\pi)$ where the first input is pair of graphs, the second input is a bijection between the sets of vertices of the two graphs, and the relation is satisfied if and only if π is an isomorphism between G_1 and G_2. Then the language associated with R is the graph isomorphism language.

A proof of knowledge for a relation $R(\cdot,\cdot)$ is an interactive protocol where a prover P convinces a verifier V that P knows a y such that $R(x,y)$, where x is a common input to P and V. The prover can always successfully convince the verifier if indeed P knows such a y. Conversely, if P can convince the verifier with reasonably high probability, then it knows, at least "implicitely," such a y, that is, such a y can be efficiently computed given x and the code of P. The precise definition follows.

Definition 10 (Proofs of Knowledge) *A proof of knowledge system for an NP relation R is a triple of polynomial time randomized algorithms P (prover), V (verifier) and E (knowledge extractor) satisfying the following properties:*

- *For every x and y such that $R(x,y)$, when $P(x,y)$ interacts with $V(x)$, V accepts with probability 1.*
- *For every prover strategy (of arbitrary complexity) P' and every x such that $V(x)$ accepts with probability at least $1/2+\epsilon$ when interacting with $P'(x)$, $E^{P'}(x)$ outputs a y such that $R(x,y)$ with probability at least $\Omega(\epsilon^{O(1)})$.*

Notice that if (P,V,E) is a proof of knowledge system for a relation R, then (P,V) is an interactive proof system for the language associated to R.

A proof of knowledge system is *zero knowledge* if it is zero knowledge when thought of as an interactive proof system.

Theorem 11 *The GIR relation has a zero knowledge proof of knowledge system.*

Proof: We use the same proof system we described in the proof of Theorem 8. We have already analysed completeness and zero knowledge, so we just need to address the existence of a knowledge extractor.

Suppose that for two graphs G_1, G_2, P' is a prover strategy that convinces the verifier V with probability at least $1/2+\epsilon$. This means that, with probability at least 2ϵ over the random choices of the prover, the prover is able to correctly answer both of the possible questions asked by the verifier in the second round. We randomly pick randomness for the prover, and then simulate the executions of the protocol with such prover randomness in the case in which the verifier sends the $b=1$ message and the $b=2$ message. With probability at least 2ϵ, we get a graph G and two permutations π_1 and π_2 such that π_1 is an isomorphism between G_1 and G, and π_2 is an isomorphism between G_2 and G. From this, it is easy to reconstruct an isomorphism between G_1 and G_2. $\Box$

4 The Quadratic Residuosity Problem

For an integer N, we say that a, $0 \leq a \leq N-1$ is a *quadratic residue mod* N if there is an r (a *square root*) such that $a \equiv r^2 \pmod{N}$.

The quadratic residuosity problem is, given N, a, to decide if a is a quadratic residue mod N. The square root relation $SQR((a,N),r)$ is satisfied if r is a square root of a mod N. Clearly the quadratic residuosity problem is the language associated with SQR.

In the following, we will restrict ourselves to the case in which a and N share no common factor, that is $a \in \mathbb{Z}_N^*$.

Let Q_N be the set of quadratic residues in $\mathbb{Z}_N^*$; our first observation is that this is a *subgroup* of $\mathbb{Z}_N^*$. The number 1 is clearly a quadratic residue (it's its own square root), if $a \equiv r^2$ is a quadratic residue, then so is its inverse $a^{-1} \equiv (r^{-1})^2$, and if $a \equiv r^2$ and $b \equiv t^2$ are quadratic residues then so is their product $ab \equiv (rt)^2$.

Consider now the following interactive protocol between a prover P that knows N, a, and a square root r of A, and a verivier V that knows N and a.

- P picks at random $t \in \mathbb{Z}_N^*$ and sends $y \equiv t^2$ to V
- V picks at random $b \in \{0,1\}$ and sends b to P
- If $b == 0$, then P sets $z \equiv t$, otherwise P sets $z \equiv rt$. P sends z to V
- If $b == 0$, then V accepts iff $z^2 \equiv y$; If $b == 1$ then V accepts iff $z^2 \equiv ya$.

We shall argue that this is a perfect zero knowledge proof system for quadratic residuosity and a zero knowledge proof of knowledge system for the square root relation.

Completeness. If all parties follow the protocol, then the verifier accepts with probability 1;

Soundness. Suppose that P' is a prover strategy that makes the verifier accept with probability $> 1/2$. Then one of the possible first messages y sent by the prover P' must be such that V accepts for both choices $b = 0$ and $b = 1$. Let z_0, z_1 be the third-round messages sent by P' in such cases. Then we have $y \equiv z_0^2$ and $ya \equiv z_1^2$, so that $a \equiv (z_0^{-1} z_1)^2$ and so a is a quadratic residue.

Knowledge Extractor. The knowledge extractor is based on the previous observation. If P' is a prover strategy that makes V accept with probability at least $1/2 + \epsilon$, then, for at least a 2ϵ fraction of the random choices of P', V accepts both for $b = 0$ and for $b = 1$. The knowledge extractor picks random choices for P', runs P' on those random choices against a V that picks $b = 0$ and a V that picks $b = 1$. If both executions are successful, and y is the first-round message and z_0, z_1 are the second round messages, then the knowledge extractor finds the square root $(z_0^{-1} z_1)$.

Simulator. Let V' be an arbitrary verifier strategy. Given N, a, the simulator algorithm for V' does the following

1. Pick at random $b \in \{0,1\}$ and $z \in \mathbb{Z}_N^*$; pick randomness R for V'.
2. If $b == 0$, set $y \equiv z^2 \pmod{N}$; else set $y \equiv a^{-1} z^2 \pmod{N}$.

3. If V', using randomness R, given y as first message, outputs b, then halt and output transcript: "V' selects randomness R, P sends y at first round, V' sends b at second round, P sends z at third round, V' accepts."

4. Go to 1

The main step in the analysis of the simulator is to observe that, regardless of the choice of b, the simulator chooses y as a unifomly distributed quadratic residue in $\mathbb{Z}_N^*$. This means that y and b, as random variables, are statistically independent, and so that the second message of V' given y is also statistically independent of b. No matter what the V' algorithm is, then, the simulator has probability $1/2$ of outputting a simulation in each attempt, and so the average number of attempts is just 2. It then remains to observe that, conditioned on a transcript being given in output, the distribution of the transcript is identical to the distribution of actual transcripts of the interaction between V' and P.

5 The Hardness of Finding Square Roots

We conclude by proving that, if $N = pq$ is the product of two distinct odd primes, then finding square roots for random quadratic residues in $\mathbb{Z}_N^*$ is as hard as factoring N.

Let A be an algorithm that, given N, and given a generated by picking $r \in \mathbb{Z}_N^*$ and setting $a \equiv r^2 \pmod N$, finds a square root of a with probability at least ϵ.

Consider the following algorithm

```
• Algorithm FACT( integer $N$ )
    – Pick a random $r \in \mathbb{Z}_N^*$
    – $t = A(r^2 \bmod N, N)$
    – Return $\gcd(t + r \bmod N, N)$
```

We will prove the following result.

Theorem 12 *Let $N = pq$ be a product of two distinct odd primes such that, for a random r, $A(r^2 \bmod N, N)$ finds a square root of $r^2 \bmod N$ with probability at least ϵ. Then FACT(N) finds a factor of N with probability at least $\epsilon/2$.*

We being by arguing that, with probability at least $\epsilon/2$, A finds a square root that is different from both r and $-r$. First, we need the following result.

Lemma 13 *Let $N = pq$ where p, q are distinct odd primes. Then every quadratic residue in $\mathbb{Z}_N^*$ has at least four square roots.*

Proof:[Of Lemma 13] Let a be a quadratic residue and r be a square root. Let $r_p := r \bmod p$ and $r_q := r \bmod q$. Note that, because a, and thus r, are in $\mathbb{Z}_N^*$, neither r_p nor r_q can be zero. By the Chinese remainder theorem, we can find $r_2 \in \mathbb{Z}_N^*$ such that $r_2 \equiv -r_p \bmod p$ and $r_2 \equiv r_q \bmod q$; we can find $r_3 \in \mathbb{Z}_N^*$ such that $r_3 \equiv r_p \bmod p$ and $r_3 \equiv -r_q \bmod q$; and we can find $r_4 \in \mathbb{Z}_N^*$ such that $r_4 \equiv -r_p \bmod p$ and $r_3 \equiv -r_q \bmod q$. Furthermore, the four numbers r, r_2, r_3, r_4 are all distinct, and they are all square roots of a. □

We can now prove Theorem 12. Let $a \in \mathbb{Z}_N^*$ be a quadratic residue such that $A(a, N)$ is a square root of a, and consider the behavior of FAC(N) conditioned on the random choice of the algorithm being an r such that $r^2 \equiv a \pmod N$. Then, under this condition, r is equally likely to be one of the (four or more[2]) square roots of a, and so, with probability at least $1/(4-2) = 1/2$ it is different from both $A(a, N)$ and from $-A(a, N)$.

Now imagine, as a random experiment, that the randomness of the algorithm is chosen by first choosing what $r^2 \bmod N$ should be like, and then choosing an r consistent with that choice. This is equivalent to just choosing r uniformly at random as in the algorithm. But this way of thinking of the choice makes it clear that there is a probability at least $\epsilon/2$ that we pick an r such that

- $t := A(r^2 \bmod N, N)$ is a square root of $r^2 \bmod N$
- $t \not\equiv r \bmod N$
- $t \not\equiv -r \bmod N$

From the first property, we have $t^2 \equiv r^2 \pmod N$, that is, $t^2 - r^2 \equiv 0 \pmod N$, and so the product $(t + r) \cdot (t - r)$ is a multiple of N. From the other two properties, however, we see that neither $(t + r)$ nor $(t - r)$ is a multiple of N, so this means that the factors of N are split between $(t + r)$ and $(t - r)$, and by computing the greatest common divisor between N and, say, $t + r$, we are going to find a non-trivial divisor of N.

[2]Actually, every quadratic residue in $\mathbb{Z}_N^*$ has *exactly* four square roots, but the lower bound that we have proved in Lemma 13 is all we need.