Notes on Complexity Theory Last updated: November, 2011

Lecture 20

Jonathan Katz

1 Zero-Knowledge Proofs

(The following notes just sketch what was covered in class.)

In the complexity-theoretic setting of interactive proofs for some language L, we are concerned about a cheating prover who might try to fool a verifier into accepting an incorrect statement $x \notin L$; there are no "security guarantees" for the prover against the (possibly cheating) verifier. In cryptographic settings, however, it may be desirable to ensure some notion of privacy for the prover. Specifically, the prover may be willing to convince the verifier that $x \in L$ but be unwilling to reveal any *additional* information to the verifier.

A "made up" example is the case of a prover who wants to convince a verifier that some mathematical statement is true (and that the prover has a proof of that fact), but does not want to reveal the proof to the verifier for fear that the verifier will then rush to publish the proof on its own. A more realistic example might be the following: say a voter casts a vote by encrypting the vote using the public key pk of some central authority, resulting in some ciphertext C. The voter might be required to *prove* that it voted honestly — that is, to prove that C is an encryption of either 0 or 1 — but the voter does not want to divulge its actual vote. To accomplish this, the parties involved can define the language

$$L \stackrel{\text{def}}{=} \{(pk, C) : \exists b \in \{0,1\}, r \in \{0,1\}^n \text{ s.t. } C = \mathsf{Enc}_{pk}(b; r)\};$$

the voter then proves to the verifier that $(pk, C) \in L$.

We discussed the following results:

1. We defined the notion of *honest-verifier* (perfect) zero knowledge, defined $\mathcal{HVPZK}$ as the class of languages having honest-verifier zero-knowledge proofs, and showed that graph isomorphism is in $\mathcal{HVPZK}$. (Note that from a complexity-theoretic viewpoint we may allow the honest prover to be computationally unbounded, but for cryptographic purposes we want the prover to be efficient.)

2. We then introduced the notion of (dishonest-verifier, perfect) zero knowledge, defined $\mathcal{PZK}$ as the class of languages having zero-knowledge proofs, and showed that the same protocol for graph isomorphism satisfies this stronger notion.

3. If we want to reduce the soundness error to 2^{-n} by repeating the graph-isomorphism protocol n times, *parallel* repetition preserves honest-verifier zero knowledge, but *sequential* repetition appears necessary if we want to preserve (dishonest-verifier) zero knowledge.

4. $\mathcal{PZK}$ cannot be too powerful; it is known that $\mathcal{PZK} \subseteq \mathbf{AM} \cap \mathsf{coAM}$. (In fact, this holds even if we only require the simulated transcript to be statistically close to the real transcript.) Motivated by this, we consider two relaxations of the notion:

- *Computational* zero knowledge requires the simulated transcript to be (only) computationally indistinguishable from the real transcript. Let $\mathcal{CZK}$ denote the set of languages having computational zero-knowledge proofs. We showed a computational zero-knowledge proof for graph 3-colorability under the assumption that (statistically binding) commitment schemes exist, implying $\mathcal{NP} \subset \mathcal{CZK}$ under the same assumption. In fact, $\mathcal{CZK} = \mathcal{IP}$ under this assumption as well.

  Statistically binding commitment schemes can be constructed from the (minimal) cryptographic assumption of one-way functions.

- A different relaxation is to require perfect zero knowledge as before, but to only demand soundness against *computationally bounded* cheating provers. In this case we refer to *arguments* instead of *proofs*. Statistical zero-knowledge arguments for all of $\mathcal{NP}$ can be based on the existence of statistically hiding commitment schemes. These, in turn, were recently shown to exist based on the assumption of one-way functions.