Lecture 4

Monday, February 23, 2009
1:26 PM

Zero-Knowledge Proofs

Figure 4.1,

Motivation: Examples, file backup, one-way function, hard-core function.

"Is it possible to prove a statement without yielding anything beyond its validity ? "

"Zero-knowledge proofs are proofs that yield nothing (i.e. 'no knowledge') beyond the validity of the assertion."

The notion of a proof:

" A proof is whatever convinces me."

Shimon Even, answering a student's question in his graph algorithms class. (1978).

- A static object v/s an interactive proof:

Traditionally in mathematics a "proof" is a fixed sequence of statements that either are "self-evident" or are Derived from previous statements via "self-evident" rules.

Here to be more precise, "self evident" should be replaced with "commonly agreed"

Examples: Axioms and derivation rules in logic.

1. Proofs are viewed as fixed objects in Maths.
2. Proofs are considered at least as fundamental as their consequences (i.e. the theorems).

In real-life situations proofs have a dynamic nature i.e. they are established via an interaction (like in a court of law etc.)

Example of NP.

Completeness and Soundness properties:

Not every set of statements has a "reasonable" proof system. Godel's incompleteness theorem and Turing's undecidability of the halting problem.

Discuss "knowledge gain"

Discuss "knowledge" v/s "information"

Interactive Proof System:

Definition 4.2.1 (An interactive Machine)

Definition 4.2.2 (Joint computation of two ITMs)

Convention : <A,B>(x), Definition 4.2.3 (Time Complexity of an Interactive Machine), (time independent of the messages received and its own coin tosses).

Definition 4.2.4 (Interactive Proof Systems). No resource bounds are placed on the computing power of the prover.

Example of NP, Exercise 2, Only languages in NP have IP systems in which both verifier and prover and deterministic.

In general IP systems could be bidirectional and verifier could be probabilistic.
Definition 4.2.5 (The Class IP) NP $\subseteq$ IP, BPP $\subseteq$ IP. Not known BPP $\subseteq$ NP ?

Definition 4.2.6, Proposition 4.2.7, Proof Exercise 1.

Graph Non-Isomorphism (GNI) in IP:

GNI not known to be in NP $\cup$ BPP. GI is in NP.

Construction 4.2.8, Proposition 4.2.9

Zero-Knowledge Proofs

Perfect and Computational Zero-Knowledge: Present definition in 4.3.1., Trivial case, BPP has perfect zero-knowledge proof system, this definition too strict, don't know of any non-trivial example in which this definition is satisfied.

Def 4.3.1 , Failure probability can be bounded above by 2^{-p(n)} for any p(n). Ex 6, statistical difference can be made negligible, Ex 8,

Computational zero-knowledge: Computationally distinguishable ensembles, Def 4..2 , In this case allowing failure of simulator with probability say 1/2 does not increase the power of Def. 4.2, Ex. 9.

Zero-knowledge actually property of prover P.

Def 4.3.3 (Alternative formulation of zero-knowledge). Two definitions equal due to universal quantification on V^*, Ex 10. One direction easy.

Analogous alternative formulation can also be obtained from Def 4.3.1 and is clearly equivalent to it.

Complexity classes PZK, CZK.

Example: Graph Isomorphism (GI) in PZK.   Construction 4.3.8, Proposition 4.3.9.

Exercises : 1, 5, 6,8,9,10.

Zero-Knowledge with respect to Auxiliary input.: Definition 4.3.10,

- The verifier, the simulator, and the distinguishing algorithm, all run in poly time in |x| and not in |x| + |z|.
- Having allowed the distinguishing algorithm to run in poly time in (|x| + |z|) would have collapsed computational zero-knowledge to perfect zero-knowledge. Ex 11.
- Def. 4.3.10 refer to computational zero-knowledge. A formulation of perfect zero-knowledge with respect to auxiliary input is straightforward. The PZK proof presented for GI is in fact PZK with auxiliary input.
- In general a ZK proof can be extended to a ZK proof with auxiliary input if in the original proof, the simulator used the verifier as a black-box. All simulators presented in this book have this property.

- Implicit non-uniformity in the Definition 4.3.10. Note that the distinguisher's time bound can be determined after the time bound on Simulator is fixed. Hence the definition 4.3.10 is robust against distinguishers which are non-uniform poly-size circuits. Present argument using universal circuit-evaluating algorithm.
- Definition 4.3.2 itself has some non-uniform flavor. A fully uniform definition would only require that it is infeasible to find the x's on which the simulation would fail. (not that such instances do not exist at all).
- Discuss fully non-uniform formulation.
    - This formulation does not guarantee an effective transformation from verifiers to simulators. Hence the level of security is unsatisfactory.
    - It does not guarantee a relation between the size of the non-uniform part of the verifier and the simulator. In Def. 4.3.10 the only non-uniform part is the auxiliary input which is the same for verifier and simulator.

  - Both the above issues arise while trying to prove sequential composition for a non-constant round of iterations of zero-knowledge proof systems.
  - The oversimplified version does not imply the basic version Def 4.3.2. For example a prover that on common input x sends some hard-to-compute ploy(|x|) bit long string that depends only on |x| (e.g. the prime factorization of all integers in the interval $[2^{|x|} + 1, \ldots, 2^{|x|} + |x|^3]$).

Sequential composition of ZK proofs :

- ZK proofs using Definition 4.3.2 are not closed under sequential composition.
- Sequential composition is an important tool in the design of ZK-proof systems. Are useful in increasing gap between the completeness and soundness probabilities.

Lemma 4.3.11 and proof. Claim 4.3.11 and proof. Claim 4.3.11.2 and proof.

What about parallel composition ? Cannot prove, in fact there are (auxiliary input) ZK protocols which can be "shown to lose ZK property when composed in parallel twice. "Non-closure" under parallel repetition also happens in many other cases like computationally sound proofs, proofs of knowledge, and multi-prover proof systems.

No Assignment this time, Ex 11 for next time.

Zero-Knowledge proofs for NP : Commitment Schemes : Two-phase (commit phase and reveal phase) two-party (the sender and the receiver) (efficient) protocols

1. Secrecy (or hiding): At the end of the first phase the receiver does not gain any knowledge of the sender's value, even when receiver tries to cheat.
2. Unambiguity (or binding): The receiver can later (i.e in the second phase) accept at most one value, as a legal "opening" of the commitment, even when the sender is cheating.
3. Viablity: If both parties follow the protocol, then at the end of the second phase, the receiver gets the value committed by the sender.

Note that w.l.o.g the reveal phase may consist of merely letting the sender send, to the receiver, the original value and the sequence of coin tosses that it has used in the commit phase. The receiver will accept the value if and only if the supplied information matches its transcript of the interaction in the commit phase.

Def 4.4.1: The secrecy requirement is a computational one, whereas the unambiguity requirement has an information-theoretic flavor and is sometimes referred to a perfect or absolute. Def 4.4.1 is computationally hiding and perfectly binding. We may have a dual definition as well namely : perfectly hiding and computationally binding.

Discuss Canonical reveal phase. Viability requirement is implicitly satisfied by this definition.

Construction based on any one-way permutation: Construction 4.4.2 (Simple bit commitment), Proposition 4.4.3.

Construction based on any one-way function : This is weakest possible assumption Ex 13. Construction 4.4.4. (Bit commitment under general assumptions), Present motivating discussion, Proposition 4.4.5 and proof.

Extensions: Sting commitment, commitment of ternary values is easy. Discuss Thm 4.4.6: Secrecy with respect to poly-size circuits.

Zero-Knowledge Proof for Graph-Coloring: Presenting a ZK proof system for one NP-complete language implies the ZK-proof system for any language in NP. This intuitive statement does require a proof though which we discuss later.
Graph 3-coloring (G3C). Motivating discussion and abstract description of the ZK-proof for G3C. Confidence in the validity of the proof can be increased by repeating in sequence the proof polynomially many times.

Construction 4.4.7 and Proposition 4.4.8

No exercises this time as well :)