CS174 Fall 98: Lecture Note 15

Michael I. Jordan, May 2002

Zero-knowledge proofs

A zero-knowledge proof is an interactive proof in which a Prover tries to convince a Verifier that she knows a secret, without actually revealing that secret.

There are many applications of interactive proof systems in computer science. In one important application, the Prover is allowed to have unlimited computational power while the Verifier is limited to polynomial time. This setup defines the complexity class IP, which turns out to occupy an interesting position in the complexity hierarchy.

In cryptography, we are interested in a setting in which both the Prover and the Verifier are limited to polynomial time complexity. This is because we are often interested in taking the idea of an interactive proof literally, and in actually implementing an interactive proof system using a communicating pair of computers.

Consider the *authentication* problem, in which you wish to convince someone that you are who you say you are. Suppose, for example, that you have published a key (p, g, y), where p is a large prime, g is a generator and $y = g^x \bmod p$, where x is a secret. You could convince someone that you are the person with public key (p, g, y) by handing over x, but in doing so you've revealed x, and the person you've given x to can subsequently impersonate you.

How might you convince someone that you know x, without actually giving that person the value of x? How might you convince someone that you know x, without giving away *any information* about x, even in repeated trials?

A zero-knowledge proof for graph coloring

Suppose that the Prover claims to know a 3-coloring for a given graph $G = (V, E)$. Can she convince the Verifier that she knows such a 3-coloring without giving away any information to the Verifier about how to color the graph?

Consider the following interactive protocol:

Protocol:

1. Prover randomly permutes the 3-coloring.
2. Verifier produces a random string r.
3. Prover *commits* the permuted 3-coloring by computing $z_i = f(c(v_i), r, k_i)$ for each i, where $c(v_i)$ is the permuted color of vertex v_i, k_i is a random key, and f is a one-way function. Prover sends all of the values z_i to Verifier.
4. Verifier chooses an edge (v_a, v_b) at random.

5. Prover sends the keys k_a and k_b to Verifier.

6. Verifier decrypts z_a and z_b and checks to see whether $c(v_a)$ and $c(v_b)$ are different. If they are different, the Verifier accepts, otherwise the Verifier rejects.

Claim: The Verifier runs the Protocol kn^2 times and accepts if each pass accepts. The probability of accepting a bad proof is bounded above by e^{-2k}.

Proof: If G is not 3-colorable, there is at least one edge whose adjacent vertices have the same color. The probability that the Verifier asks to see an edge whose adjacent vertices have the same color, and thus rejects, is at least $1/\binom{n}{2}$. Thus:

$$\begin{aligned} \Pr[\text{Verifier accepts } |G \text{ is not 3-colorable}] &\leq \left(1 - \frac{1}{\binom{n}{2}}\right)^{kn^2} \\ &\leq \left(1 - \frac{2}{n^2}\right)^{kn^2} \\ &\leq e^{-2k}. \end{aligned}$$

Note that if the graph G is 3-colorable, then the Verifier always accepts. Thus we have a one-sided error that is bounded above by an exponentially-decaying probability. The Verifier can let k be large enough to be convinced that with high probability the Prover does know a 3-coloring.

Claim: The Protocol is zero-knowledge.

Proof (sketch): We won't actually prove this claim here, but we sketch the basic idea. At each step the Verifier learns that (the Prover knows) that the pair of vertices sampled have different colors. But given that the colors are permuted independently at each step, the Verifier has no way to accumulate information across trials. That is, the information that the Verifier receives about the coloring could be *simulated* by a Simulator that simply chooses a pair of different colors for the exposed vertices at each step. The distribution across these colors is no different from the distribution obtained from the Prover.