Theoretical Cryptography, Lecture 1

Instructor: Manuel Blum
Scribe: Ryan Williams

January 18, 2006

1 Information

The course website is `http://www.cs.cmu.edu/ ryanw/crypto/`.

This class will be run a little differently from others. In particular,

- Talking during class is OK, in fact it is strongly encouraged.
- HW: Try to do it yourself. You can collaborate with others, but you must say where your ideas came from.

2 Introduction to Zero-Knowledge

For the first two weeks of the course, we will focus on a fascinating and counter-intuitive subject: *zero-knowledge*. In a real sense, zero-knowledge proofs show how to prove a theorem so that no one else can claim it. More precisely, zero-knowledge proofs can convince someone (a verifier) that a theorem is true, in such a way that the verifier cannot turn around and convince anybody else that it is true. So one can say that "zero knowledge" about the actual proof of the theorem is communicated.

How on earth can this be done? In the standard way in which we view proofs, this is impossible. We think of a rigorous proof of a theorem as a self-contained mathematical object, independent of its author. (It does not matter who or what is communicating the proof, it is still just as true and convincing.)

To develop zero-knowledge proofs, we will conceptually change the model of what it means to prove something, but not to a terribly unreasonable degree. We have two parties, a *prover* (the person who knows the theorem and its proof) and *verifier* (the proof-checker who wishes to be convinced). They will participate in an *interactive probabilistic proof.* What this means is:

1. **The prover and verifier communicate over a number of rounds.** In standard mathematical proofs, only one kind of interaction occurs: the prover gives the verifier a proof, and the verifier checks it. In zero-knowledge proofs, the prover and verifier will communicate

several times, back and forth. The proof becomes a *conversation* that ends with the verifier being convinced (if all goes well).

2. **Coins get tossed.** The prover and verifier will both be assumed to have access to randomly chosen bits, and this will play a major role in the process. The fail-safe guarantees that a formal proof system normally enjoys (*e.g.* soundness and completeness) will now only hold with some probability, albeit this probability will approach 1 as the number of interaction rounds increases. What we get in return for this sacrifice is the ability to prove theorems without giving away any information about the proof itself.

3 Example: Hamilton Cycle

We start with a motivating example: the *Hamilton Cycle* problem. Here, we are given a graph $G = (V, E)$ whose vertices are numbered $1, \ldots, |V|$, and we wish to find a cycle that visits each vertex exactly once.

The input to our zero-knowledge proof system will be (G, k), where k is a positive integer. It is useful to think of k as a *security parameter*: as k increases, the verifier's confidence in the proof will also increase.

3.1 Properties of the Proof System

In particular, we will give a proof system that takes k rounds, and has three key properties:

1. The probability that a dishonest prover $\tilde{P}$ "successfully cheats" over all the rounds is at most $1/2^k$. More precisely, if the prover does not know a Hamilton cycle, then the prover can only pass the verifier's tests with probability at most $1/2^k$. Thus the verifier has a reasonably high chance of discovering the cheating, if enough interaction takes place.

   Recall the notion of *soundness* in a formal proof system, which roughly speaking says: "If you can prove it, then it is true." The (equivalent) contrapositive of this is: "If it is not true, then you cannot prove it." In the above, we are saying that, if that the prover does not know a Hamilton cycle, then the prover only convinces the verifier *with low probability*. This is sometimes called *partial soundness*.

2. If the prover knows the proof of the theorem, our proof system will have the property that he/she can always convince the verifier of it. This is analogous to the notion of *completeness* in a proof system: "If it is true, then you can prove it." This is sometimes called *perfect completeness*, in that completeness holds with probability 1.

3. *No knowledge about the proof is communicated to the verifier.* This is by far the subtlest and most surprising point. Essentially what we mean is that if our proof communicates anything to the verifier, *the verifier could have communicated that to herself*, independently of the prover. We will analyze this sort of property in detail later.

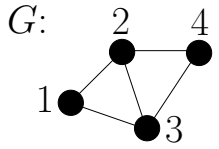
3.2 How to convince someone there is a Hamilton cycle, without giving away the cycle

We now describe the protocol that the prover and verifier follow. We will use an informal and "physicalist" description of how the protocol goes, just to motivate the ideas. Our setup will be slightly different from Manuel's lecture– we hope that it helps visualization.

Let $n = |V|$. We will describe how one round of the proof system goes. To do multiple rounds, the protocol is just done multiple times.

Privately, the prover chooses a random permutation of $\{1, \ldots, n\}$. He takes a large piece of cardboard, and draws an n by n grid on it. (This grid will correspond to the adjacency matrix of the graph G.) The prover labels the rows and columns of the matrix according to the random permutation. (He writes these numbers on the outer boundary of the grid.)

To illustrate as we go along, let's consider a fixed four-node graph G and its corresponding grid:



	1	2	3	4
1	0	1	1	0
2	1	0	1	1
3	1	1	0	1
4	0	1	1	0

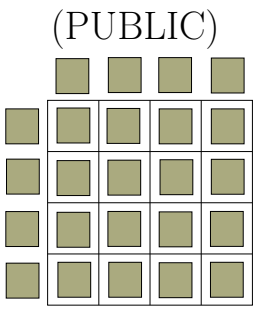
Next, the prover looks at the graph G. For each edge $\{i, j\} \in E$, he finds the row numbered i (by the permutation), and the column numbered j, and writes a 1 in their corresponding grid square. For grid squares that do not correspond to an edge, the prover writes a 0. So, now on the cardboard is a randomly permuted adjacency matrix of G.

For example, the above grid may now look like:

(PRIVATE)

	3	4	1	2
3	0	1	1	1
4	1	0	0	1
1	1	0	0	1
2	1	1	1	0

Finally, the prover paints over each row/column label and the inside of each square, with gray scratch-off paint, like what you see in lotteries. The verifier is shown the painted cardboard



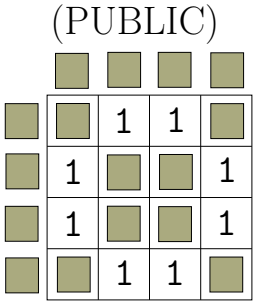
and the prover asks:

Do you want to see the graph, or do you want to see the cycle?

The verifier is required (by the protocol) to flip a coin to decide which one. (If she does not follow the protocol, then all bets are off.) Let's consider her two possible answers.

1. **"I want to see the graph."** In this case, the prover just hands over the entire cardboard, and lets the verifier scratch off everything. What the verifier sees is a randomly permuted adjacency matrix of G. Since the nodes are labelled with numbers, it is easy to efficiently check that the labelled graph on the cardboard is the labelled graph G. The verifier accepts this round, *i.e.* passes the prover, iff the cardboard graph is indeed G.

2. **"I want to see the cycle."** In this case, the prover scratches off $2n$ grid squares, corresponding to *the edges of the Hamilton cycle*. Moreover, the scratching is symmetric: if the i, j entry of the grid is scratched, the prover must also scratch the j, i entry. Nothing else is scratched off. In our example, the prover would show:



   The verifier checks that each scratched-off square has a 1 in it, and each row and column of the grid has precisely two scratched-off squares. The verifier passes the prover iff these checks succeed, as the verifier is looking at a Hamilton cycle in the underlying graph (whatever that is).

This completes the description of the protocol.

3.3 Analysis of the protocol

We would like to make three claims about the aforementioned protocol.

1. If the prover knows a Hamilton cycle, the verifier always passes him (provided he follows the rules of the protocol!).
2. If the prover does not know a Hamilton cycle, the verifier will not pass him with probability at least 1/2.
3. No knowledge about the Hamilton cycle is communicated to the verifier.

Let's analyze these claims one-by-one.

1. The first claim holds, since if a Hamilton cycle is known then the prover knows where to show its edges, regardless of the permutation. (Note the "see-the-graph" part of the protocol does not require knowledge of the cycle: if the rules are followed, then the verifier passes.)
2. Why does the second claim hold? If the prover does not know the cycle, how can he cheat? The prover could plant a new Hamilton cycle into the graph that he does know (adding edges here and there), and that would pass the "see-the-cycle" part. But then he would fail the "see-the-graph" part. If he does not plant a Hamilton cycle that he knows, then he will fail the "see-the-cycle" part.

   Therefore, if the prover does not know a Hamilton cycle, then the prover could possibly pass one of the tests, *but not both*. So the verifier will pick a test that he cannot pass with probability at least 1/2. Notice that, by repeating the protocol k times, the probability that the prover is caught in a lie increases, from 1/2 to $(1 - 1/2^k)$. (If the prover draws up 20 pieces of cardboard, and the verifier passes him on each piece, the prover is cheating with probability at most $1/2^{20}$, one in a million!)
3. Finally, we move to the third claim. What does the third claim even mean, that "no knowledge" is communicated? We argue that the verifier could have actually *simulated* the prover: for every observable thing that the prover is showing her, the verifier could have generated each of those observable things herself, with an identical probability distribution. Why is that?

   Consider the "see-the-graph" part of the protocol. What does the verifier actually see? The verifier sees the graph G written in an adjacency matrix, with its nodes randomly permuted. But the verifier has access to random bits– she could have generated a random permutation and drawn up that adjacency matrix herself, with identical probability for each permutation!

   Now think about the "see-the-cycle" part of the protocol. The verifier sees $2n$ grid squares with 1's in them, and there are two for every row and column. Since the prover chose a random permutation of node labels, the distribution of cycles that the verifier sees is exactly the same distribution as that which one would see from choosing a uniform random cycle on n nodes. (Why? This is *not* obvious. You must convince yourself why this is true.) Hence, if

the verifier chose a random n-node cycle, drew up the adjacency matrix for it, and painted over all the 0's– this would produce the same distribution of objects.

Therefore, the verifier could have sat back and randomly generated all the observables that she is viewing, with an identical probability distribution. She flips her coin– if it comes up heads she generates a random permutation of G, if it comes up tails she generates a random cycle.

So in a real sense, anything that the verifier learns from the prover, she could have learned on her own. Still, if the prover always passes her tests, he is convincing, because he always meets her challenges (see the graph, see the cycle) no matter which she asks for. The fact that the prover cannot predict the verifier's coin is absolutely crucial– if he could do that, he could always fool her. Because he cannot predict her coin, he must somehow prepare for both tests in advance, or risk being found out. But the only way to pass both the "see-the-graph" and "see-the-cycle" tests is to know a Hamilton cycle in G!

Of course, everything we have been doing has been at a considerably high level. Over the next few lectures, we will begin to formally define what a zero-knowledge proof system is, and learn how to prove more theorems in zero-knowledge.