15-503/859P: Cryptography **Lecturer:** Manuel Blum
Topic: Zero-Knowledge Proofs **Date:** 25 Jan 2006
Scribe: Yinmeng Zhang

3.1 Overview

1. The first homework is out on the website; it is due next wednesday (1 Feb) at the beginning of class. We will discuss an incorrect answer to problem 3 (vertex cover).
2. Definition of Zero-Knowledge Protocol
3. Review the ϕ-Knowledge Protocol for the Subset Sum Variant from last lecture, and discuss improvements.

3.2 Historical Note

During the Cold War, the Americans and Russians agreed to certain guidelines for atom bomb tests. Both sides wanted to be able to check that the other was following the rules; they could accomplish this by burying a signaller at the other country's testing site. Of course, both sides also wanted to ensure that the signaller wasn't sending more information than absolutely necessary. Further, it wasn't feasible to allow the other country to examine the signaller before it was buried, for fear of sabotage. In other words, what was desired was a **zero knowledge proof that the buried signaller worked correctly**. The countries engaged in a simple one round protocol. The Prover provided *two* signallers. The Verifier then picked one to examine; if the Verifier was satisfied with the signaller, then the *other* signaller would be buried.

3.3 Vertex Cover

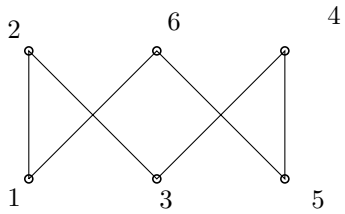
Recall that a problem is in NP if there exist short proofs of yes instances, e.g. for Hamiltonian Cycle a proof that a graph really had a cycle would be exhibiting the cycle.

In HW1#3, we discuss another NP problem, **Vertex Cover**. An instance of the Vertex Cover problem consists of

1. Input: a graph $G = (V, E)$ and a positive integer Bound B
2. Output: YES if G contains a set $S \in V, |S| \leq B$ such that all edges are incident to some vertex in S (that is, for any edge (u, v), either $u \in S$ or $v \in S$); NO otherwise.

We can picture a vertex cover as a King's Crown: the cover is the base, and there can be no edges between the points of the crown.

Example: consider the hexagon or 6-cycle. The hexagon with bound $B = 3$ is a YES instance (we can exhibit the cover $S = \{1, 3, 5\}$), but the hexagon with $B = 2$ is a NO instance - no two vertices can cover all the edges.



3.4 Definition

A zero knowledge protocol for a given Decision Problem Π is a set of instructions for Prover P and Verifier V having as inputs

1. an instance of Π
2. a security parameter k

with the properties

1. **(Completeness)** If the instance is a YES-instance and P knows a proof, then P can *efficiently* follow the protocol.
2. **(Guarantee to an *honest* Verifier)** If the Verifier is honest (follows protocol), then a dishonest prover $\tilde{P}$ (one who does not actually know a proof) ~~cannot fool the Verifier~~ can fool the Verifier only with exponentially small probability. That is, with probability at most $1/c^k$, for some positive constant c greater than one, where k is our security parameter.
3. **(Guarantee to an *honest* Prover)** If P is honest then even a dishonest Verifier $\tilde{V}$ will get "no information" from P that she could not have gotten without the prover. That is, if $\tilde{V}$ has an efficient algorithm $\mathcal{A}^P$ to compute something (anything at all, not just the original proof), then $\tilde{V}$ could compute the same efficiently without P - particularly, V could *simulate* P.

3.4.1 What do we mean by "efficient"?

Efficient means in polynomial time with respect to the size of the input. This is why it is important to have a security parameter k - its size represents how many rounds we are willing to run. Technically, because we make use of randomness, we are talking about a *probabilistic polynomial time* algorithm, which is simply a usual polynomial time algorithm except that we allow coin flips as steps.

3.4.2 What do we mean by simulate?

The $Verifier$ must be able to produce the correct distribution of P's output. In terms of the two-headed coin analogy, either half of the coin must be easy to fake, but both simultaneously should be very hard.

3.5 Incorrect Vertex Cover ϕ-Knowledge Protocol

Consider the following incorrect protocol for Vertex Cover: given an instance $G = (V, E)$, let the Prover construct a lottery ticket containing the graph G with labeled vertices. The Verifier may ask the Prover to either

1. show the vertices and the edges
2. show the edges and POINT to the vertex cover (without revealing the vertex names)

Clearly the algorithm is efficient. (1) confirms that we are working with the correct graph and (2) allows us to check that every edge is incident to at least one vertex in the vertex cover, so we also satisfy the guarantee to the verifier.

However, the proof is not zero knowledge, because revealing (2) reveals information about the degrees of the verticies in the vertex cover. Particularly, imagine a graph which contains exactly one vertex of degree c (where c is any number). Then (2) will reveal whether or not that vertex is in the vertex cover.

3.6 Subset Sum Variant - a concrete example

Recall the subset sum variant discussed last time:

Let n be some positive even integer. An instance consists of

1. n positive n-bit numbers, $\langle a_1, \ldots, a_n \rangle$
2. a positive n-bit integer A

The question is

> Does there exist $\langle x_1, \ldots, x_n \rangle$ with $x_i \in \{0, 1\}$ s.t. $\sum_{i=1}^{n} x_i = n/2$ and $\sum_{i=1}^{n} a_i x_i \pmod{2^n} = A$?

Let's run the protocol from last time with $n = 4$. Our instance will be

1. $\langle 3, 7, 13, 8 \rangle$
2. 5

where an honest prover *privately* knows that $13 + 8 = 5 \pmod{16}$.

To make a "lottery ticket", the prover chooses a random string of r's, say

> 7,7,3,9

and derives a remainder string of s's

12,0,10,15

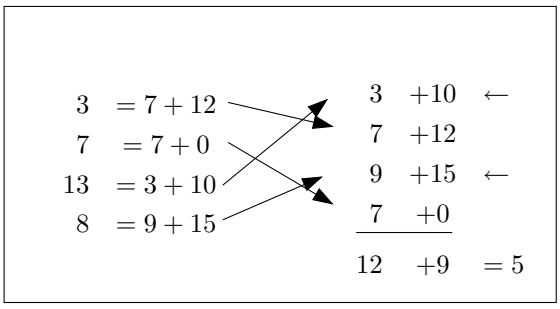
The prover then chooses a random permutation, say

(2,4,1,3)

to rearrange the r and s columns and points arrows at the appropriate rows. Finally, the prover writes down the partial sums

$R = 12 \qquad S = 9$

and covers everything up with "grey paint".



The Verifier can then choose to view one of the following:

- private1 := the r, s split [check problem statement]
- private2 := r column and R and arrows [check rcol sum]
- private3 := s column and S and arrows [check scol sum]
- private4 := R and S [check $R + S = A$]

Note that instead of grey paint, we can think of the hidden information as being stored in a number of envelopes, which the Verifier can ask the Prover to open. However, there is a subtlety here. For this protocol, we would need *6* envelopes, even though there are 4 different private pieces. This is because the pieces overlaps, and we must ensure, for example, that the R revealed in private2 is the same R that is revealed in private4. Thus, in the envelope formulation, the envelopes would be (1) private1, (2) the r column, (3) the s column, (4) R, (5) S, and (6) the arrows. The Verifier may choose to view (1) only, (2) and (4) and (6) only, etc.

A lottery ticket is not hard to make if the Prover knows where the arrows should go, so we have Completeness. A completely filled lottery ticket exhibits a proof that the instance is YES, so we have the guarantee to the Verifier.

Recall that the guarantee to the Prover is that any single private piece could be generated by the Verifier with the same distribution that the Prover generates it. We can check this for each private

piece, so the guarantee holds. This implies that the Verifier will catch a dishonest Prover with probability at least 1/4 in a single round.

It would be nice to be able to have only three private pieces, which would improve the probability the verifier catches a dishonest Prover to 1/3.

3.6.1 Can we make private4 public?

One proposal is to make private4 public. Note that from private2 alone the Verifier can derive private4, since R and A are given; similarly, private3 also gives private4. If we could show something similar for private1, then there would be no point in hiding private4, and we would only have three private pieces.

Unfortunately, note if private1 and private4 are simultaneously revealed, then we have introduced two new equations into the problem, namely

$$\sum r_i x_i = R \text{ and } \sum s_i x_i = S$$

Further, recall that the r_i are random numbers - so we have gained an equation independent to the original $\sum a_i x_i = A$! In fact, each round will supply a new equation, independent to all previous. Four independent equations are sufficient to solve for the four x_i in our example, so after only 3 rounds the Verifier has learned the Prover's proof.

In general, n rounds will supply enough equations to solve for $x_i, \ldots, x_n$. So much for zero-knowledge.

3.6.2 2/3 Bound Rescued

However, there is a cute fix to this idea. We can define

- newprivate1 = oldprivate1
- newprivate2 = oldprivate2 and oldprivate4
- newprivate3 = oldprivate3 and oldprivate4

using the same lottery ticket as before. Above, we already argued that newprivate1 and newprivate2 can be simulated by the Verifier, so this formulation is in fact zero-knowledge.