YALE UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

CPSC 467a: Cryptography and Computer Security *Handout #24*
Felipe Saint-Jean and Michael Fischer *December 13, 2005*

Solution to Problem Set 8

In the problems below, "textbook" refers to *Introduction to Cryptography with Coding Theory: Second Edition* by Trappe and Washington..

Problem 36: Zero knowledge interactive proof for 3-colorability

An undirected graph $G = (V, E)$ is said to be 3-colorable if there is an assignment $\gamma : V \to \{1, 2, 3\}$ such that for all edges $\{v, w\} \in E$, $\gamma(v) \neq \gamma(w)$. The problem of testing if an arbitrary graph is 3-colorable is known to be $\mathcal{NP}$-complete. Alice claims to know a coloring γ for the public graph G.

Here is the idea for a zero knowledge interactive proof whereby Alice can demonstrate knowledge of a 3-coloring γ to Bob without revealing any information about γ. Alice generates a random permutation $\rho : \{1, 2, 3\} \to \{1, 2, 3\}$, defines a new coloring of the graph $\gamma' = \rho \circ \gamma$, commits to each of the colors $\gamma'(v)$ for $v \in V$ using bit commitment, and then sends each of the commitments to Bob. Bob picks an edge $\{v, w\}$ of G and asks Alice to reveal the hidden colors corresponding to v and w. Alice does so and Bob verifies that they are different.

(a) Explain why Bob's verification always succeeds if Alice and Bob are honest.

(b) Explain how a dishonest Alice who does not know a 3-coloring of G can fool Bob if she can correctly guess in advance which edge Bob is going to ask about.

(c) Explain why a dishonest Alice who could successfully answer any of Bob's permissible questions in fact does know (i.e., could efficiently compute) a 3-coloring of G.

(d) What is the probability that Bob will catch a dishonest Alice who doesn't know a 3-coloring of G on one round of the protocol?

(e) How many times does this protocol need to be repeated in order to make Alice's probability of successful cheating less than 10^{-6}?

(f) Explain why the protocol is zero knowledge.

Solution:

part a

If they are both honest then γ' is a correct 3-coloring of the graph. Then when Bob requests Alice to open the commitments for an edge (u, v), he will be able to check what colors where assigned to that node pair, convincing himself that u and v have different colors in γ'.

part b

If Alice knows what edge Bob will ask for she can color those two nodes with different colors and color the other nodes in any fashion. The result will not be a correct 3-coloring, but it will convince Bob when he asks for the correctly colored edge.

part c

If Alice could successfully answer any of Bob's permissible questions, this means that every edge has a consistent coloring under γ', that is, γ' colors the two endpoints differently. But then γ' is a 3-coloring of the graph, contradicting the assumption that Alice does not know a 3-coloring.

part d

An incorrect coloring has at least one edge with an inconsistent coloring. If $G = (V, E)$ then the probability of Bob choosing an inconsistently colored edge, and thereby catching a dishonest Alice, is at least $\frac{1}{|E|}$.

part e

He will need to repeat the protocol t times where t is the smallest integer such that,

$$\left(1 - \frac{1}{|E|}\right)^t \leq 10^{-6}$$

part f

In each round Bob learns the fact that two nodes have different colors in γ'. He also learns what those colors are. However, that information does not help him to piece together a complete 3-coloring of the graph from the knowledge gleaned from many rounds of the protocol. Because the permutation ρ is chosen at random each time, the coloring γ' changes on each round. This makes the pair of colors assigned to the two endpoints of an edge $\{v, w\}$ on one iteration of the protocol statistically independent of the color pair assigned to the revealed edge on any other iteration, so the information Bob gets from one round of the protocol is of no help to him in finding a complete 3-coloring. All he learns at each round is a random pair of distinct numbers. Those pairs of numbers could be computed by a simulator (which suggests how a formal proof of zero knowledge would proceed).

Problem 37: Secret sharing basics

(a) Textbook, problem 12.3.2.

(b) Textbook, problem 12.3.3.

Solution:

part a

The Lagrange interpolation polynomial that goes through $(1, 13)$ and $(3, 12)$ is

$$p(x) \equiv 13 \cdot \frac{x-3}{1-3} + 12 \cdot \frac{x-1}{3-1} \pmod{101}$$

Thus, $p(2) \equiv 13 \cdot 2^{-1} + 12 \cdot 2^{-1}$. Now $2^{-1} \equiv 51$, so $p(2) \equiv 63 \pmod{101}$.

part b

The polynomial is

$$p(x) \equiv 8 \cdot \frac{x-3}{1-3} \cdot \frac{x-5}{1-5} + 10 \cdot \frac{x-1}{3-1} \cdot \frac{x-5}{3-5} + 11 \cdot \frac{x-1}{5-1} \cdot \frac{x-3}{5-3} \pmod{17}.$$

Plugging in $x = 0$, we get

$$\begin{aligned} p(0) &\equiv 8 \cdot \frac{0-3}{1-3} \cdot \frac{0-5}{1-5} + 10 \cdot \frac{0-1}{3-1} \cdot \frac{0-5}{3-5} + 11 \cdot \frac{0-1}{5-1} \cdot \frac{0-3}{5-3} \pmod{17} \\ &\equiv 8 \cdot \frac{15}{8} + 10 \cdot \frac{5}{(-4)} + 11 \cdot \frac{3}{8} \pmod{17} \\ &\equiv 8^{-1} + 4^{-1} - 8^{-1} \pmod{17} \\ &\equiv 4^{-1} \equiv 13 \pmod{17}, \end{aligned}$$

so the secret is 13.

Problem 38: Secret sharing with cheater

Textbook, problem 12.3.6.

Solution:

This problem is subject to many different interpretations. In the version that we consider here, each participant sends its share to each other participant. The honest participants send their correct shares. The cheater can send a different share to each participant. On the basis of the received shares, each participant votes for a secret. The recovery succeeds if all honest participants vote for the correct secret. (The cheater can vote for anything.)

part a

If two participants meet and the cheater is among them then the cheater can cause any secret to be recovered by simply claiming that his share is the point that lies on the line determined by that (fake) secret and the honest participant's share. Hence, no information about the secret can be determined.

With three participants, each of the three pairs of participants can reconstruct a secret. If one of the participants is a cheater, the three reconstructed secrets may all differ, but one of them (the one from the two honest players) is correct. To find which secret is the correct combination to the safe, all three combinations can be tried in turn.

Note that each honest participant can actually narrow the set of choices to the two secrets which they helped to reconstruct, but the two honest players will not agree on what those sets are. Although they have a common intersection, they also don't know which other player is honest, so the cheater can prevent them from reaching agreement on the correct secret.

part b

As mentioned above, three participants are not sufficient for the honest participants to determine the secret. With four participants, the problem is easily solved. Each honest player reconstructs three secrets, one with each of the other participants. Two of those participants are honest, so two of the three reconstructed secrets will be the correct secret. Hence, each honest player simply votes for the majority value of these three reconstructed secrets.

Problem 39: Secret sharing implementation

Textbook, problem 12.4.3.

Solution:

After recovering all the secrets from all possible pairs,

Participants		Recovered secret
A	B	69918
B	C	927070
C	D	21502
D	A	21502
A	C	21502
B	D	391311

From the table we can conclude that the correct secret is 21502 and B has a broken share. The program used to compute the result follows.

ho23-p39.java

```
import java.math.BigInteger;

public class LagrangePolinomial {
    BigInteger _points[][];
    BigInteger _n;
    // expects and array of n,2 where points[i][0] is x_i and point[i][1] = y_i

    private LagrangePolinomial(String[][] points, BigInteger n){
        _points = convert(points);
        _n=n;
    }


    private BigInteger[][] convert(String[][] points) {
        BigInteger[][] ret = new BigInteger[points.length][2];
        for (int i=0;i<points.length;i++){
            ret[i][0]= new BigInteger(points[i][0]);
            ret[i][1]= new BigInteger(points[i][1]);
        }
        return ret;
    }
    private BigInteger delta(int j,BigInteger x) {
      BigInteger res=BigInteger.ONE;
      int i=0;
      for(i=0;i<_points.length;i++){
        if (i!=j)
            res=res.multiply(x.subtract(_points[i][0]))
.multiply(_points[j][0].subtract(_points[i][0]).modInverse(_n)).mod(_n);
      }
      return res;
    }

    private BigInteger eval( BigInteger x){
      int i=0;
      BigInteger res=BigInteger.ZERO;
      for (i=0;i<_points.length;i++){
         res=res.add(_points[i][1].multiply(delta(i,x))).mod(_n);
      }
      return res;
    }
    static void reveal(String[][] shares, BigInteger mod,String name){
        LagrangePolinomial pol = new LagrangePolinomial(shares, mod);
        System.out.println(name+ " & " + pol.eval(BigInteger.ZERO) + "  \\\\");
    }
    public static void main(String[] args){
BigInteger n = new BigInteger("984583");
        String[] a = new String[]{"38","358910"};
        String[] b = new String[]{"3876","9612"};
        String[] c = new String[]{"23112","28774"};
        String[] d = new String[]{"432","178067"};

        reveal(new String[][]{a,b},n,"AB");
        reveal(new String[][]{b,c},n,"BC");
        reveal(new String[][]{c,d},n,"CD");
        reveal(new String[][]{d,a},n,"DA");
        reveal(new String[][]{a,c},n,"AC");
        reveal(new String[][]{d,b},n,"BD");
    }
}
```