Massachusetts Institute of Technology
6.915: Network and Computer Security
Professor Ronald L. Rivest

Handout 20
October 19, 1995

Problem Set 3 Solutions

Problem 3-1. Birthday Paradox

Suppose we choose k values $s_1, \ldots, s_k$. Let X_{ij} denote the following random variable

$$X_{ij} = \begin{cases} 1 & \text{if } H(s_i) = H(s_j) \\ 0 & \text{if } H(s_i) \neq H(s_j) \end{cases}$$

Because of the random oracle assumption we know that

$$Prob[X_{ij} = 1] = 2^{-m}$$

which means that $E[X_{ij}] = 2^{-m}$. So if we denote by X the total number of collisions among $s_1, \ldots, s_k$ we have by linearity of expectations that

$$E[X] = \Sigma_j \Sigma_i E[X_{ij}] \approx \frac{k^2}{2^{m+1}}$$

Now we need to choose k such that $E[X] = 1$ and by solving the above we get $k \approx 2^{\frac{m}{2}}$.

Problem 3-2. Secret Sharing

(a) Consider a vector of t shares $(s_{i_1}, \ldots, s_{i_t})$. This vector gives no information about the secret so all v secrets are possible. However this vector can be extended to a $t+1$ vector in w possible ways, and a $t+1$ vector identifies the secret exactly. So if $w < v$ then it would not be possible for $(s_{i_1}, \ldots, s_{i_t})$ to generate all possible secrets and as such this vector would contain information about the secret

(b) There are many solutions to this problem.

The first one assumes only that the number of good trustees is at least $t+1$ and it is only computationally secure. At sharing time the dealer publicizes a public key E and signs the shares with the corresponding secret key $D = E^{-1}$. When trustees reveal their shares they are required to reveal the signature as well. If the signature scheme is secure and the bad trustees don't have enough computational power to forge signatures, then good trustees will be able to detect bad shares and eliminate them from the pool of shares. Since there are at least $t+1$ good trustees, they will be able to reconstruct the secret.

There is an unconditional secure version of the above solution. It works if we assume that the bad trustees are at most $n/2$. The idea is that the dealer will give trustee T_i the following numbers:

- the share s_i

- $n-1$ keys $k_{i,1},\ldots,k_{i,n}$.
- $n-1$ MACs of the share s_i with the keys held by the other trustees. $MAC_{k_{1,i}}(s_i),\ldots,MAC_{k_{n,i}}(s_i)$ (these are the one–time unconditionally secure MACs we saw in class)

When T_i reveals his share he has to reveal all the MACs too. Then trustee T_1 can check that T_i is revealing the right share because he has $k_{1,i}$, so can T_2 and so on ... If say T_3 is not happy with the share revealed by T_i he votes against him. If more that half of the trustees vote against T_i he is disqualified and his share thrown away. At the end we are left with only good shares that allow us to reconstruct the secret.

Another solution works for Shamir's secret sharing scheme. It assumes that the bad trustees are at most t and that $t<\frac{n}{3}$. Its security holds unconditionally. The solution is based on the so–called Reed–Solomon error correcting codes. Recall that the n shares $s_1,s_2,\ldots,s_n$ are created as $s_i=f(i)$ where f is a polynomial of degree t. Such shares can be thought as an *encoding* of the polynomial f. Such encoding is of course redundant since only $t+1$ numbers would be sufficient to encode f. The idea behind Reed–Solomon's codes is that such redundancy is useful in error-correction. Indeed if at most $n/3$ of the s_i values are incorrect or missing it is still possible to recover the whole polynomial f and thus the secret.

Problem 3-3. Zero–Knowledge proof for discrete logarithms

Repeat the following 20 times.

Alice chooses a random number $r \bmod p-1$ and sends to Bob $w=g^r \bmod p$

Bob sends a challenge $b\in\{0,1\}$ to Alice

Alice replies with $z=r+bx \bmod p-1$

Bob checks if $g^z=wy^b \bmod p$, if it does he accepts otherwise he rejects.

Problem 3-4. Public–Key Certificates

This was again an open–ended kind of question. The most common answers were the following:

For Approach 1: people noticed the the verification procedure is simple and allows for an easy key revocation/changing mechanism for users. On other hand it is potentially slow and overloads the root of the system.

For Approach 2: It is potentially more efficient (possibly shorter paths), it distributed, if the root is temporarily compromised it is not as bad as in approach 1. Cons: key changing/revoking is more complicated, each internal node has to store more stuff.