POIS-6

24/01/2023 (Tue)

Today:

- Reduce DLP to MSB
- Design a provably secure pseudo random generator

MSB Problem

Given that $y = g^x \bmod p$. Find whether $x < \frac{p-1}{2}$

If we can show that there is an efficient algorithm to find the MSB, we can find an efficient algorithm to solve the DLP problem.

Algorithm for solving DLP using MSB(x)

First, we calculate the LSB of x. basically calculating whether x is odd or even.
Case 1: x is even
Calculate $y' \equiv \sqrt{y} \bmod p \equiv g^{\frac{x}{2}} \bmod p$
Case 1: x is odd
Calculate $y' \equiv \frac{y}{x} \bmod p \equiv g^{x-1} \bmod p$

So, essentially if x is odd, we change the last bit to zero. And if we have x as even, then we knock off the last bit. We keep repeating this process over and over again, until we get all the bits. So what is the problem of this method ?

The issue is that y has two square roots, $g^{\frac{x}{2}}$ and $g^{\frac{x}{2}+\frac{p-1}{2}}$. Only the former can knock off the last bit. Hence, if we instead get the other square root, we will not be able to knock off the square root. Hence we are stuck.

This is where we require the MSB problem, if we are able to figure out the MSB problem, then we can find out whether $x < \frac{p-1}{2}$, meaning that this is the solution that we require, otherwise we reject the exponent.

Hence, if we can solve MSB, we can solve DLP. Thus we now know that MSB is a hard-core predicate for the DLP problem

Calculating the square root modulo of z if p is 3 modulo 4

Just calculate $z^{(p+1)/4}$, this will give one of two square roots of z.

Example
$z = 9$, $p = 19$
z has two square roots, 3 and 16
$z^{\frac{(p+1)}{4}} \equiv 9^5 \bmod 19 = 16$

Designing a secure pseudo random generator

Notice that the DLP problem, can be thought of as a permutation (for every x there is a unique $y = g^x$)

Given a one-way permutation f and it's hard core predicate h, we define the pseudo random generator $G : \{0,1\}^n \to \{0,1\}^{n+1}$

$$G(x) = f(x) \mid\mid h(x)$$

Assume that there exists a distinguisher D which can distinguish between the uniform distribution of n bits, and the string created by G.

$$|P\left[D(U_{l(n)}) = 1\right] - P\left[D(G(U_n)) = 1\right]| > negl(n)$$

Obviously the first n bits, in the random distribution and G have identical distribution, both are just $f(x)$, hence we only need to show the last bit is indistinguishable. Also, the last bit is also almost random (probability 1/2) since it is a hard core predicate, therefore even if we have have information about $f(x)$, there is no way to figure out the value of $h(x)$. Hence the complete message is completely indistinguishable, hence we have a valid pseudo random generator.

Notice that this pseudo random generator can only increase the length of strings by 1. Therefore if we want to increase the length of strings by an arbitrary amount, then we can just keep performing the same pseudo random generator over and over again to generate a string of the given length (no clue for the proof)

Chosen Plaintext Attack

Can be used to crack One-Time Pad

Recall that the adversary definition of perfect secrecy is that when presented with two messages , and an encryption of one random message between the two, the adversary should not be able to figure out the original message, m_0 or m_1 (probability is 1/2)

However, in the case of chosen plaintext attack, the adversary also has access to the encryption algorithm in detail, that is, they can pass any message they want through the algorithm and see the output.

We notice that the One-Time Pad is not CPA secure, because, it is very easy to figure out the index of the message (0 or 1) : just pass both the messages through the encryption tool, and compare the output of both, with the encryption of the random message. Hence the One-Time Pad is not CPA secure

In fact, no deterministic encryption scheme is CPA-Secure, because the idea is still the same, we have the encryption algorithm, and since the encryption tool generates the same encryption every time (since it is deterministic), he can just generate the encryption of the two messages, and compare and figure out the original message.

Will discuss Probabilistic Encryption in next class.