CIS 3362 11/20/2024

1. Strong Collision Resistance/Weak Collision Resistance
2. Birthday Paradox
3. Homework 7 – go over

Weak Collision Resistance: Given a fixed output, y, for a hash function, it should be computationally infeasible to find a value x for which H(x) = y.

Strong Collision Resistance: It should be difficult to find ANY to values x_1 and x_2 such that $H(x_1) = H(x_2)$ (GOAL)

Birthday Paradox

We'll assume that all 365 birthdays are equally likely and leap year babies don't exist.

There are n people in a room and the probability that at least 2 of them share the same birthday is greater than 50%. What is the smallest possible value of n?

This is the same as the strong collision resistance requirement.

There are D days total in a year, each one equally likely for a birthday. There are n people in the room.

(a)What is the probability that all n people have different birthdays?

(b) What is the probability that at least 2 of the people share the same birthday?

First person has a new birthday with probability D/D.

Second person has a new birthday with probability (D-1)/D.

Third person has a new birthday with probability (D-2)/D.

Ans for QA = $\frac{D}{D} \times \frac{(D-1)}{D} \times \frac{(D-2)}{D} \times \ldots \times \frac{(D-n+1)}{D}$

(b) Ans for QB = $1 - \frac{D}{D} \times \frac{(D-1)}{D} \times \frac{(D-2)}{D} \times \ldots \times \frac{(D-n+1)}{D}$

$$1 - \frac{D}{D} \times \frac{(D-1)}{D} \times \frac{(D-2)}{D} \times \ldots \times \frac{(D-n+1)}{D} > \frac{1}{2}$$

$$\frac{D}{D} \times \frac{(D-1)}{D} \times \frac{(D-2)}{D} \times \ldots \times \frac{(D-n+1)}{D} < \frac{1}{2}$$

Here is code

```
class Main {

    final public static int DAYS = 365;
    final public static int NUMROWS = 50;
    public static void main(String[] args) {

        double probUnique = 1;
        for (int num=DAYS; num>DAYS-NUMROWS;
num--) {

            probUnique = probUnique*num/DAYS;
            System.out.println( (DAYS-num+1)+".
"+(1-probUnique));
        }
    }
}
```

For D = 365, minimum value of n is 23.

Elliptic Curve Equation

$$y^2 \equiv (x^3 + ax + b)\ (mod\ p)$$

For a fixed value of x, there is either 0, 1 or 2 solutions.
First, plug in x into RHS and reduce mod p this will give you something like

$$y^2 \equiv c\ (mod\ p)$$

where 0 <= c < p

This will have either 0, 1 or 2 solutions for y.

It has 1 solution only if c = 0 → y =0.

Otherwise, we need to do two things:

(a) Determine if this has 0 or 2 solutions.
(b) If it has 2 solutions, find them.

If $p \equiv 3\ (mod\ 4)$, tasks (a), (b) are is easy.

Calculate

$$c^{\frac{p-1}{2}}\ (mod\ p)$$

We learned before via Fermat's Theorem, that this number has to be either 1 or p – 1 (-1). Thus, What we're going to is calculate this number. If this number is equal to 1, then we know we have 2 solutions.

If this number is equal to p – 1, then hit continue and go to the next value of x.

<u>Part B: Finding those two matching values of y.</u>

This part is easy only when $p \equiv 3\ (mod\ 4)$. Here is one value of y that works:

$$y \equiv c^{\frac{p+1}{4}}\ (mod\ p)$$

Why does this work?

$$y^2 \equiv (c^{\frac{p+1}{4}})^2\ (mod\ p)$$

$$y^2 \equiv c^{\frac{p+1}{2}}\ (mod\ p)$$

$$y^2 \equiv c^{\frac{p-1+2}{2}}\ (mod\ p)$$

$$y^2 \equiv c^{\frac{p-1}{2}+\frac{2}{2}}\ (mod\ p)$$

$$y^2 \equiv c^{\frac{p-1}{2}}c^1\ (mod\ p)$$

Previously, we calculated $c^{\frac{p-1}{2}}$, it's equivalent to 1 mod p

$$y^2 \equiv (1)c^1\ (mod\ p)$$

$$y^2 \equiv c\ (mod\ p)$$

To get the other valid value of y, just do p – y, where y is equal to our first solution above.

How do we encode a fixed set of plaintext bits in a point?

Now we know that for a value of x, we can test if tehre's a solution for y. The problem is NOT ALL values of x are valid. But, about half of them are, and which ones are valid is pretty random.

So about half of all x's work…and we know how to test if one works.

Let's say is 32 bits.
Consider sending 24 bits. Put the bits you send in the least significant spots.

M =11001100 11111111 01010101

There are $2^{32-24} = 256$ bitstrings of the form

Xxxx xxxx 11001100 11111111 01010101

Randomly plug in different values of the first 8 bits and test to see if that value of x works. The probability that all 256 don't work is only $½^{256}$.