MATH 314 Fall 2019 - Class Notes

11/20/2019

Scribe: Jake Wines

Summary: Today's class covered cryptographic hashes and what a good cryptographic hash hopes to accomplish, along with the introduction of birthday attacks and the statistics behind them.

Cryptographic Hashes

It should be hard to find collisions in a cryptographic hash. There are 3 resistances that a cryptographic hash should prioritize:

- Preimage Resistance - It should be hard to imagine the output of a hash given an input.
- Weak Collision Resistance - It should be hard to find x hash given y input (A specific hash, a specific input).
- Strong Collision Resistance - It should be hard to find any x hash given any y input (Any hash, Any input).

Discrete Log Hash

1. Find a large prime q where p = 2q+1 is also a prime.
2. Find two different primitive roots, alpha and beta (mod p). These should be random.
3. Since alpha and beta are both primitive roots (mod p), there exists a variable a such that: $alpha^a \equiv beta(modp)$.
4. This presents the discrete log problem. Finding a solution should be hard.

Hashing Algorithm

Hash messages of size at most $q^2 - 1$. $m < q^2$
Produce a digest smaller than p. This should be roughly equivalent to m^2.
Write m in "base q".

- $m = x_1 + x_2 q$
- $0 <= x_1, x_2 < q$

$h(m) \equiv alpha^{x_1} * beta^{x_2} (modp)$

The discrete log hash isn't practical, however. SHA-128 or SHA-256 is used instead.

Birthday attacks

Question: How many people do you need in a room before two people share a birthday? What's the probability that two people share a birthday? 1/365.
For three people, the probability that all three people have a different birthday is $(1-1/365)*(1-2/365)$.
For k people, the probability is $(1-1/365)(1-2/365)...(1-(k-1)/365)$
Which turns out to be roughly equivalent to $e^{-k^2/(2)365}$ (We're measuring for the probability of no collision here)
The probability of a collision would therefore be $1 - e^{-k^2/2N}$ where N is the number of days.

So for example, how many cars are needed before the probability of the last 3 digits of two liscense plates being the same is greater than 50 percent? We're assuming that no letters are involved here, so we have 10 digits, 0-9.
Therefore, $N = 10^3 = 1000$.

- $1 - e^{-k^2/2(1000)} < 1/2$
- $ln(1/2) < -k^2/2(1000)$
- $2000ln(1/2) < -k^2$
- $-2000ln(2) < -k^2$
- $2000ln(2) > k^2$
- $sqrt(2000ln(2)) > k$
- k is roughly equal to 37.

We want the probability of finding collisions in our hash functions to be small.
Alice has a signature function and a hash function. Her hash function produces digests with 60 bits. Alice signs her message m using s(h(m)).
Eve drafts two contracts: One good, one bad.
She finds 35 places in each contract where she can make a small change that doesn't affect the meaning of the contract.
She has 2^{35} different good contracts and 2^{35} different bad contracts.
She takes all of these contracts and computes the hash of all of them: 2^{36} in total.
There exists 2^{60} total possible digests given that Alice is using 60 bits.
Given the birthday problem, we can compute $e^{-(2^{36})^2/2(2^{60})} = e^{-2^{72}/2^{61}} = e^{-2^{11}} = e^{-2048}$ which is rougly equivalent to 10^{-890}, or basically just zero.

So, a good contract and a bad contract have the same hash.
Eve gives Alice the good contract to sign, M_g. Alice signs it , $S = S(h(M_g))$. But since $h(M_g) = h(M_b)$, Alice's signature is also valid for the bad contract.
To avoid this scenario, Alice simply changes the message slightly before signing it, therefore invalidating the signature for the bad contract that Eve had planned on swapping out for the good one.