MATH 314 Spring 2018 - Class Notes

5/8/18

Scribe: Brennan Traube

Summary: Finished discussion on digital signatures and how to mathematically tie it to a person. Then discussed the birthday paradox (not really a paradox).

Notes: In order to have a signature that is mathematically tied to both the message and the sender, we first use a cryptographic hash function to produce a digest of the message, then sign the digest.

- Send(Original Message, S(h(m))) = (m,r)... m is the message and r = S(h(m))
- If Bob receives (m,r) he checks if r is a valid signature for h(m)

Examples: If signing using RSA then:

- S(h(m)) = $h(m)^d(modn)$ d is the "Signature creator"
- Bob checks the signature (m,r) by checking whether $h(m) \equiv r^e(modn)$
- RSA key is (n,e) with decryption exponent d

Notes: Birthday Paradox. How many people do you need in a room before the probability that two people share a birthday is greater than 50 percent?

- If you sample 23 people, there will be at least a 50 percent chance that two of them share a birthday
- The probability that 2 people share a birthday is (1/365)

In order to make sense of the probability of more people sharing a birthday, we should calculate the probability that they DON'T share a birthday. The following calculations will be out of 366 days, in order to include leap years.

The probability that 3 people don't share a birthday is: $(366/366) * (365/366) * (364/366)$

For K people, the probability that no two people share a birthday is: $(366 * 365 * 364 * ...(366 - K + 1)/366^K)$

Notes: In general, if you have N things being chosen randomly lambda times, then the probability that none of the N things is picked twice is:

$$e^{(-\lambda^2)/2N}$$

Notes: Birthday attack on digital signatures

- Suppose Alice is using a hash function h(x) that produces digests with 50 bits
- Suppose h(x) is pre-image resistant and weekly collision resistant.
- Alice uses h(x) to sign her message.
- Eve wants to trick Alice into signing a bad contract
- She (Eve) writes a good contract that Alice would be willing to sign.
- She (Eve) finds 30 places where she can make a small change to the contract
- Alice should be happy to sign any of the 2^{30} contracts
- Eve repeats this process for bad contracts, which Alice would not agree to sign
- Eve now has $2^{30} + 2^{30} = 2^{31}$ total contracts
- Eve will compute the hash of every single one of these contracts
- 2^{50} digests exist

  $N = 2^{50}$
  $\lambda = 2^{31}$
  Now, using the equation from earlier, we get:

$$e^{-((2^{31})^2)/(2*2^{20})} = e^{-2048}$$

  This number means that there exist a lot of collisions between the good and the bad contracts.

- Eve goes to Alice with a good contract which, when hashed, produces the same has as a bad contract.
- Eve then takes Alice to court later and presents the judge with the bad contract and its hash (Alice's signature on it.

Notes: How to defend against this attack?

- Alice could make digests with bigger bit size

- Alice can change something small about the contract before signing (Example: adding a comma)
- GENERAL PRACTICE: Never sign something you didn't produce yourself or make a small change to