MATH 314 Spirng 2020 - Class Notes

5/6/2019

Scribe: Matthew Clark

Summary: In todays class we talked about hash functions. This consisted of breifly going over why they are used and and how they worked. We then went over the discrete log hash and the process of using it

Notes: Include detailed notes from the lecture or class activities. Format your notes nicely using latex such as

- Last class: discussed digital signatures. These gave us a way to produce a signature that could be produced by alice (requreis her private key)
- Also depends on the message m, but isnt as quite tied to message as it is as alice
- How do we sign a message that is larger than the modulus? (n-RSA) or (q DSA)
- Dont want to break our message up into lots of pieces and sign each piece individuality
- One solution is just to take the entire message modulo n first and sign that.
- Unfortunately lots of messages will have the same signature $m, m + n, m + 2n$) all have the same signature
- Even though only Alice can produce a valid signature for m eve could replace M with $M' = m + n$ or m + k for any k and the signature S that alice produced for M would be valid for the messages too. S is a valid signature for many different messages
- Any time two messages that have the same signature that is a collision.

Hash Functions

- -A hash function is a function is a function h(x) that takes in a large input and returns a much smaller output called a digest.
- If two different inputs produce the same output that is called a collision. Every hash function has collisions. Pigeon-hole-principle
- Goal: Make collisions hard to find —— Ex: h(x) = x(mod2)
- Digest are small h(3) = h(5) collision. This is not a good cryptographic hash function
- Same if we take h(x) = x(mod x)
- Even if n is large and is easy to find collision:
- h(1) = h(n+1) = h(2n +1)

-
- Never going to eliminate collision
- Ideal properties of cryptographic hash function
    1. Preimage resistance: given a digest it should be hard to find an input x where h(x) = y
    2. Weak collision resistance: given an input x is should be hard to find an input x2 with h(x1) = h(x2)
    3. Strong collision resistance: It should be hard to find any two inputs x1 and x2 with h(x1) = h(x2) and (x1 = x2)
- What is the difference between weak and strong collision resistance?
- Weak are input fixed
- Strong get to pick both messages
- SHA-1 is no longer strongly collision resistance

<u>Discrete log hash</u> - Probably strongly collision resistance

- Proof showing if we find a single collision we can use the collision to solve a difficult discrete problem.
- since solving discrete log problem is hard finding any collision must also be hard
- Setup: pick a large prime q and p where q = 2p
- Ex: p = 11 q = 23
- Find two different primitive roots (mod q): u and B
- Since they both primitive roots powers of then produce everything (mod q)
- In particular u = B^c(mod q) for some c, but finding c is hard
- If $B^[x] = 1($ mod q)
- X = 0(mod q-1)
- q-1 divides x
- Discrete log hash takes in inputs between 0 and p^2-1 output digest between

- 0 and q-1 since:
- q = p and $q^2 = p^2$
- Hash function is producing digest about square root the size of inputs
- Not arbitrary large inputs.
- How do we compute h(x)?
- - Write x in base p, x = a, p a_0 —— $0 < a_0, a_1 < p$
- Since $x < p^2$ we know there will be two digits
- H(x) = h(a,P + a0) = $u^{a_0} * B^a (modq)$
- This hash has lots of collision n p^2 inputs 2p+1 outputs Each output has $p^2/(2p+1)$ = p different preimages
- Each digest has about p different collisions
- Even though there are lots of collisions finding one is really hard
- Suppose Eve manages to find one collisons to the discrete log hash.
- H(x1) = H(x2) x_1 doesnt = x_2
- X_1 = a1p + a_0 —— $X_2 = B_1 * p + B0$
- Then $h(x_1) = h(x_2)$ $alp^{a_0} * B^{a_1} = alp^{b_0} * B^{b_1}$(mod q)
- $alp^{a_0 - b_0} * B^{a_1 - b_1}$ = 1(mod q)
- now we said: alp = B^c(mod q) for some c but finding c is hard (DLP)
- $(B^c)^{a_0 - b_0} * B^{a_1 - b_1}$ = 1(mod q)
- This exponent is divisible by (q-1)
- So c($a_0 - b_0$) + $a_1 - B_1$ = 0(mod q-1)
- q = 2p +1 q-1 = p
- c($a_0 - B_0$) = $a_1 - B_1$ (mod 2p)
- (mod p) we can solve this c = $(b_1 - a_1)(a_0 - B_0)^{-1}$(mod p)
- Not 0 since x1 and x2 are different
- Guess either c = 0 (mod 2) or c =1 (mod 2)

- Use the Chinese remainder theorem to find c(mod 2p)
- That means we found the c that makes u = b^c(mod q)
- supposed to be hard
- Therefore it should be hard to any collisions to the discrete log hash
- Unfortunaley we dont us DL hash much in practice
- - Cant use arbituary large number s
- - Much slower than Sha-2 etc
- To sign a message alice want tpo use noth a digital signature s(x) proving that it came from her and cryptographic secure hash function h(x)
- How does she use both?
- 2 options (m, s(h(x))) or (m, h(s(m))
- Sign the hash dont hash the signature