MATH 314 Spring 2018 - Class Notes

11/26/2018

Scribe: Jude Barlow

Summary: This class covered hashing and the birthday paradox.

Digital Signatures: Way for Bob to "sign" a digital message to prove that it came from him.

- Digitally the problem is that there isn't a good way to put a signature that is "attached" to the message like a physical signature is.
- We need a mathematical way to connect a signature and a message. To do this, we use a hash function.

Hash Functions:

A hash function is a function that takes in a "large" (arbitrarily large) input and produces a much smaller output. The output of a hash function is called a digest.

Examples: Parity Bit

$h(m) \equiv m \pmod 2$
Digest 0 or 1
We could do this for any modulus N.
$h(m) \equiv m \pmod N$

- Easy to find a message with the digest 1 for example
- Hash functions are NEVER one-to-one. This is the pidgeon-hole-principle.
- Any time that two different messages have the same output is referred to as a collision of the hash.
- Every hash function has collisions. Cryptographically we would like these collisions to be hard to find.

Ideal Properties of a Cryptographic Hash:

1. Preimage Resistant: given a digest y it should be hard to find an input x such that $h(x) = y$.
2. Weakly-Collision Resistance: given input m1, it should be hard to find another message m_2 that has a collision with $m1$; $h(m2) = h(m1)$.

3. Strongly-Collision Resistant: It should be hard to find any two messages m1 and m2 where $h(m_1) = h(m_2)$; $(m_1) \neq (m_2)$

A Strongly-Collision-Resistant Hash:

Discrete Log Hash

Discrete log problem given
$y = a^x \pmod{p}$
knowing $y, a, p...$ it is hard to find p.

Pick a large prime p where
$q = 2p + 1$ is also a prime
$p = 3, 5, 11$

Find two primitive roots
$\alpha, \beta \pmod{q}$
to hash a message $m < p^2$.
take m and write it in base p
$m = a_1 + a_2 p$
$0 \leq a_1 \leq p$,
$0 \leq a_2 \leq p$

$h(m) = \alpha^{a_1} * \beta^{a_2} \pmod{q}$

Inputs: $0...p^2 - 1; q = 2p + 1$
Outputs: $0...q - 1; q^2 = (2p + 1)^2 \Rightarrow q^2 \cong 4p^2 \Rightarrow p^2 \cong \frac{a^2}{4}$

- Outputs are about square root of the size of input.
- Decent hash function in terms of producing smaller digests.
- Lots of collisions but I claim this has strong collision resistance.

Suppose I have a way of finding collisions for this hash...
This means I can find messages m_1, m_2 where $h(m_1) = h(m_2)$
$m_1 = a_1 + a_2 p$
$m_2 = b_1 + b_2 p$
$h(m_1) = h(_2) \Rightarrow \alpha^{a_1}\beta^{a_2} \equiv \alpha^{b_1}\beta^{b_2} \pmod{q}$
$\alpha^{a_1 - b_1}\beta^{a_2 - b_2} \equiv 1 \pmod{q}$
Recall: α and β are both primitive roots mod q this means there is some number we can raise α to get
$\beta; \alpha^c \equiv \beta \pmod{q}$

Discrete log problem: Finding c should be hard.

Using $\beta \equiv \alpha^c \pmod{q}$, we get
$\alpha^{a_1-b_1} \times (\alpha^c)^{a_2-b_2} \equiv 1 \pmod{q}$
$\alpha^{(a_1-b_1)+c(a_2-b_2)} \equiv 1 \pmod{q}$
This means that $(\alpha_1 - \beta_1) + c(a_2 - b_2)$ is a multiple of $(q-1) = 2p$

This means $(a_1 - b_1) + c(a_2 - b_2) \equiv 0 \pmod{2p}$
$c(a_2 - b_2) \equiv (b_1 - a_1) \pmod{2p}$
$c = (b_1 - a_1) \times (a_2 - b_2)^{-1} \pmod{2p}$
Assuming we can find a collision of the hash, we can find "c" that is supposed to be hard to find. Finding any collision to the discrete log hash is hard.

Disadvantages of Discrete Log Hash:

1. Can't use arbitrarily large inputs.
2. It is slow.

In practice hashes like MD5 or SHA-x (neither of these are no longer recommended).

What if we attack a hash by brute force? Hash lots of messages until two have the same digest.

Birthday Paradox (Birthday Problem):

- How many people do you need to have before it is more than 50 percent likely to share a birthday?
- What is the probability 2 people share a same birthday?

$1 - (\frac{364}{365})$

- Probability of 3 people?

$1 - (\frac{364}{365})(\frac{363}{365})$

- Probability of k people?

$1 - (\frac{364}{365})(\frac{363}{365}) \times ... \times (\frac{366-k}{365})$

- How big does k have to be before this is bigger than $\frac{1}{2}$?

$k = 24$ suffices.