CS 370 Notes
week 3

Hash Functions

- function that generates k bits from n bits where $k \leq n$
- Ascii Parity bits (7 data, 1 p.b)
- 8-bit CRC
  $\hookrightarrow$ Cyclic Redundancy Check
  $\hookrightarrow$ xor all bits, send as chk
- Hash / Checksum
  $\rightarrow$ no key
  $\rightarrow$ One way
- Mac
  $\rightarrow$ hash with key so that message's authenticity is assured ~

for $h: A \rightarrow B$

- for $x \in A$, $h(x)$ is easy / efficient to calculate
- It is hard to find $y \in B$ that maps to $x \in A$ where $h(x) = y$
- Weak collision resistance: It is computationally infeasible to find an input $x' \in A$ that has the same hash as known input $x \in A$, $x \neq x'$ given x, find x'
- Strong collision Resistance: No
  two values $x, x' \in A$ can be feasibly found that produce the same hash. $\Big\}$ find $\big\{ x, x'$
- There are going to be many collisions: Pigeonhole Principle

---

- Pigeonhole Principle: maps of larger input space to smaller output space will create collisions.
- All we need is sizeof(output space) number of inputs + 1 to be guaranteed to create collision

Birthday Paradox
- If hash has m bits, $2^{m/2}$ hashes (tries) results in 50% chance of collision.
  $\hookrightarrow 2^{m/2}$ security on hashes.

Message Authentication Codes
- Can use last block of i.e. cbc
- can use $h(k \oplus m \oplus\oplus k)\| k = $key
  $\circ$ Textbook MAC
- Hmac: msg m in k blocks of b bytes
  $k' = $key and output l bytes
  $\rightarrow$ ipad = 00110110 repeated b times
  $\rightarrow$ opad = 01011100 repeated b times
  $\hookrightarrow$ hmac $-h(k', m) = h\big((k' \oplus opad \|$
  $h\big((k' \oplus ipad \| m)\big)$
- Hmac can be used with hash
  func even if the hash is vulnerable to collisions, i.e. Sha-1

Public Key Crypto

- Public key and private keys are different but related

- **Confidentiality:** enc with pub
- **Integrity & authentication,** enc with private key, also Non-Repudiation
- PKey systems are much slower than symmetric systems.
- Usually used in coordination with symmetric crypto.
- Based on the solutions of hard problems
  ↳ RSA: factoring of large primes
- RSA, Elliptical Curve Public key algorithms can be used for signing, encryption and symmetric key distribution, unlike most others.

Requirements of an Asymmetric Enc

- Computationally easy to enc, dec, generate a key pair
- Computationally infeasible to break encryption scheme
- either can be used for enc with the opposite being used for dec.

Diffie-Hellman Key Exchange

- Discrete logarithm Problem:

$$n = g^k \mod P \quad \Big|\, n, g \in \text{integers} \atop P \in \text{Primes} \Big\} 1024 \text{ bit}$$

- Everyone knows prime P and int g.
- Key is generated with random k, → n

Person A   const: P, g Person B

$$PK = \text{pub} = g^{k_a} \mod P \qquad PK = g^{k_b} \mod P$$

—private: # K_A (key share)   —Private= K_B
       pub

$$K_{AB} = (PK_B)^{K_A} \mod P \qquad K_{AB} = (PK_A)^{k_B} \mod P$$

- shared key   shared key

- Diffie Hellman Man ITM
→ if malicious attacker installs itself in the middle of the initial PK exchange, they can sit in the middle, decrypt everything and send it onwards
   ↳ susceptible because no host identification.
   ↳ Solved with PK Certs.
     that make sure PK ⟺ Person x

Totient Function $\phi(n)$

- # number of $< n$, and relatively prime to n
  ↳ Relatively Prime: no factors common with n
  ↳ $\phi(10) = 4 \leftarrow \{1, 3, 7, 9\}$

Euler's Theorem

- $X^{\phi(n)} = 1 \mod n$
- $n = p \cdot q$ (composite of primes)

RSA (Rivest, Shamir, Adleman)

— MIT 1977

Next page

- Choose two large primes p, q
- Let $n = pq$, then $\emptyset(n) = (p-1)(q-1)$
- Choose $e < n$ s.t. e is relatively
  Prime to $\emptyset(n)$
- Compute d s.t. $d \cdot e \mod \emptyset(n) = 1$

Public Key (e, n) Private Key (p, q, d)

- encipher: $m^e \mod n = $ ciphertext
- decipher: $c^d \mod n = $ message