CS370 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
  - ↳ Cyclic Redundancy Check
  - ↳ xor all bits, send as chk
- Hash / checksum
  - → No key
  - → One way
- Mac
  - → hash with key so that message's authenticity is assured

for $h: A \to B$
- 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 infeisable 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 feisably found that produce the same hash. (find $\{x, x'\}$)
- 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 gvarenteed to create collision

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

Message Authentication Codes
- can use last block of i.e. CBC
- can use $E[h(K \,||\, m \,||\, K)] \,||\, K$ - key
  - ↳ Textbook MAC
- HMac: msg m in b blocks of b bytes (k' = key) and out l bytes
  - → ipad = 00110110 repeated b times
  - → opad = 01011100 repeated b times
  - ↳ hmac - $h(k', m) = h(k' \otimes opad \,||\, h(k' \otimes ipad \,||\, m))$
- Hmac can be used with hash fxns even if the hash is vulnera to collisions, i.e. Sha-1

NEXT PAGE

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 infeisable to break encryption scheme
- either can be used for enc with the opposite being used for dec.

Diffie-Hellman Key Exchange

- Discrete logarithm Problem:
  $n = g^k \bmod p \mid n, g \in \text{integers}, p \in \text{Primes}$ — 1024 bit
- Everyone knows prime p and int g.
- Key is generated with random K, → n

const: p, g

Person A		Person B
$PK = pub = g^{K_A} \bmod p$		$PK = g^{K_B} \bmod p$
	(key share) pub	
-private: ~~K_A~~		-private = K_B

$K_{AB} = (PK_B)^{K_A} \bmod p$ — shared key

$K_{AB} = (PK_A)^{K_B} \bmod p$ — 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

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

RSA (Rivest, Shamir, Adleman)

- MIT 1977
-

Next Page

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

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

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