

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
 - ↳ 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 $h: A \rightarrow B$

- 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.
- 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 size of (output space) number of inputs + 1 to be guaranteed 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 $[h(K \parallel m \parallel K)]$ | K = key
- o Textbook MAC
 - $K' = \text{key}$
- Hmac: msg m in n blocks of b bytes and out l bytes
 - $\text{ipad} = 00110110$ repeated b times
 - $\text{opad} = 01011100$ repeated b times
 - ↳ $\text{hmac} = h(K', m) = h(K' \otimes \text{opad} \parallel h(K' \otimes \text{ipad} \parallel m))$
- Hmac can be used with hash fns even if the hash is vulnerable 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
- PK 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.

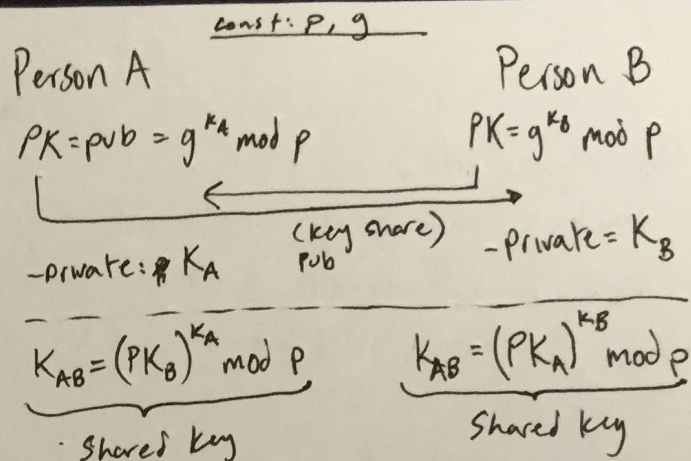
- Computationally easy to enc, dec, generate a key pair
- Computationally infeasible to break encryption scheme
- either can be used for enc with the opposite being used for dec.

Diffie - Hellman Key Exchange

- Discrete logarithm Problem:

$$n = g^k \mod p \quad \left| \begin{array}{l} n, g \in \text{integers} \\ p \in \text{Primes} \end{array} \right\} 1024 \text{ bit}$$

- Everyone knows prime p and int g .
- Key is generated with random K , $\rightarrow n$



- 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 $PK \leftrightarrow \text{Person } x$

Totient Function $\phi(n)$

- ~~A~~ 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 $\phi(n) = (p-1)(q-1)$
- 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: $m^e \bmod n = \text{ciphertext}$
- decipher: $c^d \bmod n = \text{message}$