9/19/17

CRYPTOGRAPHY: 3 properties

- SECRECY - ~~encryption~~ message can only be read by intended recipient → encryption
- INTEGRITY - message cannot be altered in transit → hash function
- AUTHENTICATION - recipient of msg can verify identity of sender → signatures

KERCKHOFF'S ASSUMPTION: cryptosystem should be secure EVEN if the algorithm is known. OPPOSITE of "security by obscurity"

↳ typically people can pretty easily figure it out

EX: copy protection algorithm

EX: putting sensitive software/date on a public-facing server, but not telling anyone the IP address of the server (nmap: port scanning)

ENCRYPTION:

- take input data (plaintext) & transform into a scrambled form (ciphertext)
  - symmetric is where both parties have the same key
    ex. AES, RC4, serpent, Blowfish

FIESTEL NETWORK * K can be anything, but should be secret

- consume a fixed-size input block & output a ciphertext block of the same size

→ in each round, divide input block into 2 halves (L & R)

→ in each round, R is unchanged, but L is replaced with the output of the round function.

→ at the end of the round, swap positions of new L & R.

to decrypt...

→ feed the ciphertext back through the network but in reverse - reverse the order of K (key for round) F(K, R)

* very easy to implement in hardware or software → x86 includes it!

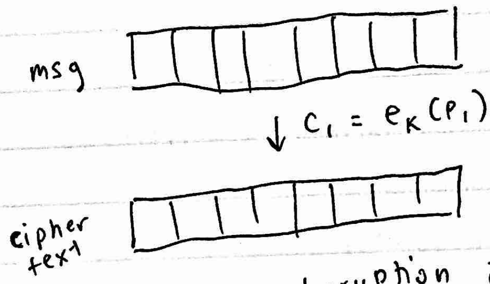
GOOD ROUND FUNCTION:

1. produces a lot ~~of~~ of diffusion: small change in input should affect output as much as possible
2. should produce a "high confusion": small change in KEY should affect ~~out~~ output as much as possible.

BLOCK CIPHERS MODE:

- ECB: electronic code book msg 

→



$\downarrow c_1 = e_K(P_1)$ encryption

decryption is just the reverse

Disadvantages: same ~~ciphertext~~ plaintext block always generates the same ciphertext block
- → like you can just analyze frequency & get the cipher
- Attacker can ~~det~~ delete/add blocks to ciphertext & recipient can't detect it

~overall, bad~

COUNTER ("CTR")

- for each message, select a random initialization vector (IV) – can be public
- split plain text into blocks

$$c_i = P_i \oplus e_K(IV + i)$$

IV is like a big number

↑ KEY is still secret

- Advantages:
  1. two plaintext blocks w/ same content will encrypt to different ciphertexts
  2. if attacker modifies/shuffles ciphertext blocks → cascading decryption errors

Solve the rendevous problem...?

Asymmetric crypto systems!

- RSA: a public key & a private key for each participant
  - small (?) problem is the host for public keys, but assume they are published somewhere
  - private key is kept secret
  - sender encrypts msg w/ ~~pr~~ public key of recipient
  - recipient decrypts msg w/ private key

EULER TOTIENT FUNCTION

$\phi(n)$: how many numbers btwn 1 & n-1 are relatively prime to n
(meaning that the number & n have no common divisors besides 1)

~ Suppose Alice wants ~~to~~ to let people ~~to~~ send her an encrypted msg

- Alice picks a modulus n, find an encryption exponent e & decryption exp d.

$$e \cdot d = 1 \bmod \phi(n)$$

Alice's public key is (n, e) * a tuple
Alice's private key is (n, d)

→

→ LogJam

- suppose that Bob has a msg M that he wants to send to Alice
  - Bob represents M as an integer, $0 \le M \le n$
  - Bob sends ~~a~~ ~~en~~ ciphertext to Alice $C = M^e \bmod n$
  - Alice decrypts ~~M~~ message by doing $M = C^d \bmod n$

    $(M^e)^d \bmod n = M^1 \bmod n = M \bmod n$

in RSA, e & d are interchangeable

9-26-17

Hash functions ü

- Take a variable-sized msg as input & ouputs a fixed-size result (btwn 8-64 bits)
- desireable properties:
  1. It should be hard to invert the function given a hash value & it should be hard to generate an input w/ that ~~#~~ hash value ("one way property")
  2. Should be rare for ~~any~~ two inputs to map to the same value ("collision resistance")
- examples: SHA family, Whirlpool, MD5 (insecure, known vulnerabilities)

Ensure the integrity of the msgs we send:
  Append a hash value for msg to the end of msg, when recipient gets msg, recipient can verify hash value.

~ Authentication ~

- Symmetric crypto: message authentication codes (MAC's) calculate a hash value over

msg	secret

  send msg + MAC to someone who knows the secret key

RSA Signatures

- calculate ~~HASH~~ hash value for msg to sign, H.

  Signed val $= H^d \bmod n$ * remember d is private, e is public
- recipient can uncover H by $H = \text{signedval}^e \bmod n$

Certificate: binds a principal (a person) to a public key

- X.509 is most popular format for certificates, it includes:
  1. issuer field ← eg. Verisign ("certificate authority), vouches for principal's identity
  2. signature algorithm. (like SHA 1 w/ RSA encryption)
  3. subject (eg: foo.com) → principal ~~for~~ to whom we are binding public key →

↓ continued...