CIS 3362 11/18/24

Hash function

Input: arbitrary size (any number of bits)

Output: fixed size k bits (2^k possible output values)

many-to-one function

good hash function makes the probability of each output value roughly equally likely + given a fixed output value it should be computationally infeasible to generate an input that has that hash value as output.

Analogy: One use passwords

WORST

Store table w/ user names + passwords.

FIRST improvement

Store username + hash(pswd)

alice 10111O... 11

Weakness: One could create a "rainbow table"

1. Make a list of potential passwords
2. For each calc hash(pswd).
3. Compare outputs from #2 with my stolen list!

IMPROVEMENT

adding "salt" to a pswd

table		
user	"salt" rnd bitstring	hash value
U1	$salt_1$	$hash(pswd_1 \| salt_1)$
U2	$salt_2$	$hash(pswd_2 \| salt_2)$

effort to creat tables is multiplied by # of users.

One big use is message authentication

K = key, M = msg, H(m) = hash value of msg

1. Alice sends Bob $E(K, [M \| H(m)])$

   $\|$ = concatenate

   Bob decrypts, gets M' and $H(m')$.
   He calculates $H(M')$ ← makes sure it matches this

2. Alice sends Bob $M \| E(K, H(m))$

   M = plaintext everyone see

   $E(K, H(m))$ = encrypt hash

   Bob decrypts to get $H(m)'$

   Makes sure $H(m) = H(m)'$ (received)

3. Alice sends Bob $M \| H(M \| S)$

   S = secret value

4. Alice sends Bob $E(K, M \| H(M \| S))$

   Version 3 w/ encryption added as last step

Verifying who sent the message

Alice → Bob

$$M \,||\, E(PR_K, H(M))$$

To verify Alice sent message.

PR_K: Alice's Private key

PU_K: Alice's Public key

Bob takes this and uses PU_K and uses this to decrypt and reveal $H(M')$ if this matches $H(M)$, Alice had to have sent it.

If we also want the message to be secret we can send

$$E(K, M \,||\, E(PR_K, H(M)))$$

K: secret shared key for symmetric system

Another equivalent but slower idea

$$E(PU_{Bob}, E(PR_{Alice}, M))$$

Bob receives this +

1. Decrypts w/ PR_{Bob}.
2. Decrypts w/ PU_{Alice}
3. If this is readable
   a) Alice must have sent ct
   b) Only Bob can read ct.

But this is extremely slow because encryption/decryption via Public Key Cryptography is much slower than Private Key + here large messages are fully being encrypted (twice) via Public Key systems