CIS 3362 11/17/25

Wed - Return Quiz 5 + Com Service (Get grades in if I don't)
Tues - Post HW DUE (Due Tues on finals week)

Today: hash function use in crypto
Different than use of hash functions in CS1

Hash function

Input: arbitrary size / type (for crypto we think of it as an arbitrarily long bitstring)
Output: fixed size (256 bits, 512 bits, etc.)
Many-to-one function
Multiple inputs could map to the same output.
For a good hash function we want:
1) the probability of each output to be roughly equally likely.
2) Given an output value y, it should be computationally infeasible to find a value of x, such that $f(x) = y$.
3) In general it should be difficult to find $x_1 \neq x_2$ but $f(x_1) = f(x_2)$.] COLLISION

Use of hash functions for passwords

BAD WAY TO ~~MANGE~~ MANAGE PASSWORDS

acct	pswd
bob01	02468
alice22	1234
⋮	

→ bad to store. If stolen all passwords are compromised!

IMPROVEMENT #1

acct	f(pswd)
bob01	11010011
alice22	10011101

→ If I steal this, I don't automatically know your password

Problem given $y=f(x)$, find one value of x s.t. $f(x)=y$.

To improve our chance of making this useful (from attacker's standpoint):

Create a Rainbow Table

pick 10^6 common pswd, for each calculate $f(x)$ + store in look up table

Improvement #2

Adding SALT

concatenate

acct	salt (rnd bitstring	$f(\text{pwd} \,\|\| \text{salt}_i)$
bob01	$salt_1$	$f(\text{bob's pw} \,\|\| \text{salt}_0)$
alice22	$salt_2$ ...	

WORK / MEMORY multiplied by n

n = # users

Use of ~~mac~~ hash functions in message authentication.

Alice $\xrightarrow{M}$ Bob (Bob wants to verify that M hasn't been altered since alice wrote it.)

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

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

Bob uses K to decrypt, gets M' and H(M'). Uses H to calculate his H(M') + double checks that it matches the second item.

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

everyone can read message but only Bob can verify that it hasn't been altered.

Bob ~~decrypt~~ decrypts to get H(M'). Bob then takes M, sees if H(m) = H(m').

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

↳ secret shared value (like a key)

Bob to check calculates $M \| S$, because he knows S then $H(M \| S)$, and sees if it matches the 2nd item sent.

4. Alice sends Bob

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

Do step 3 but encrypt whole thing!

Bob would have to 1st decrypt, then carry out steps of #3.

Goal is to authenticate the message means message hasn't been altered, but different than a digital signature.

Digital Signature is proof that someone authored something.

Allow Bob to verify that Alice sent the message

Alice → Bob

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

(Alice's private key)

remember in RSA, the public key e and private key d, are symmetric. We could encrypt w/ d and decrypt with e.

↳ Can be undone by anyone because e for Alice is public.

To make MSG secret:

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

(K: use shared secret key)

why not do whole message? Public is slow so it's faster to encrypt hash value then the whole message

---

SLOWEST WAY TO BOTH VERIFY SENDER + RECIPIENT + MSG HASN'T BEEN CHANGE

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

↓

Only Bob can undo this! Then by using Alice's public key, he proves Alice sent it!