CS370 Notes

WEEK 7

Multi Factor Authentication

- factors of authentication
  - → what you know
  - → what you have
  - → what you are (bio)
  - → what you do {handwriting, voice}
  - → where you are
- MFA must be commensorate to the value of the target being protected.

Biometric Authentication

- covers everything from finger-prints to retinal paterns (physical features)
- also covers behavioral traits (gait, voice, typing rhythm)
- More ~~m~~entropy than passwords
- what makes a good biometric?
  - → all users must have this
  - → must be unique
  - → permanence of trait over time
  - → easy to measure, collect
  - → Accuracy, robustness, acceptability
  - → Circumvention is hard
- current Biometrics:
  - → fingerprints
  - → retinal scans
  - → face ID
  - → Voice
  - → keystroke dynamics (believed to be unique)
- Biometric data collection process
  1] Collect data 2] Store in template
  3] Authentication matching template
- Performance Measures
  - → can enrollment complete / or is there an Enrollment Failure Rate
  - → Failure to Capture Rate
  - → False Positive rate & False Negatives.

Authentication vs Identification

- Survelliance (avoid detect) or Authentication (try to detect)

One Time Passwords

- can be used exactly once, then they are invalidated.
- Challenge-response mechanism used
- Problems: Password generation, distribution and synchronization between users and servers.

S/Key OTP

- based on idea of Lamport
- User chooses seed K
- server calculates

  $h(K) = K_1$; $h(K_1) = K_2 \cdots$

  Server saves $h(K_n) = \underbrace{K_{n+1}}_{\text{stored}}$

  Then, $P_1 = K_n$, $P_2 = K_{n-1} \cdots$ ($K_n = h(K_{n-1})$)

  ↳ If K_n intercepted, cannot find K_{n-1}, as hash cannot be undone...

- Server always remembers the last Password.

$$\text{user} \xrightarrow{\text{name}} \text{server}$$
$$\text{user} \xleftarrow{\{i\}} \text{server}$$
$$\text{user} \xrightarrow{\{p_i\}} \text{server}$$

server: $h(p_i) = h(K_{n-i+1}) = K_{n-i+2} = P_{i-1}$

↳ if match, store p_i where P_{i-1} used to be. then decrement i

NEXT PAGE

HOTP OTP w/ HMAC

- server and user share secret K and counter C.
- $HOTP(K, C) = Trunc(HMAC\text{-}SHA\text{-}1(K, C))$
- HOTP Password = $HOTP(K,C) \bmod 10^d$ | ~~d = len(msg)~~
  - ↳ d = len of password
- ↳ Truncate() extracts 31 bits starting at (i+1) where i is last 4 ~~digits~~ bits of MAC value.
- counter is updated after success

TOTP - Time OTP

- $C_T = \text{floor}(\lfloor (T - T_0)/\text{Time step} \rfloor)$
- time step usually is 30s, $T_0 = 0$.
- use $HOTP(K, C_T)$ to compute.
- Validator will also check $C_T + 1$ and $C_T - 1$ (to account for transit time and misconfigurations.
- How periodic pw generation (authenticator Google, ...) are generated.

Authentication Protocols

- cryptographic using protocols to allow authentication, usually running on top of other network protocols

PAP - Password Authentication Protocol
- RFC (Req for comments, IEEE) 1334 } for PPP
- unsecure, transfers pw's in clear

CHAP - Challenge Handshake Auth Proto
- replace PAP; Also has problems } also PPP

EAP - Extensible Authentication Protocol
- used on 802.1x. Many methods from EAP framework

RADIUS Auth Protocol

- centralized authentication, authorization and Accounting for network service
- + widely used; uses other Auth Proto, i.e. PAP, EAP...
- + superceded by DIAMETER

KERBEROS

- windows.

PAP DIAGRAM

NOT ENC'D

user $\xrightarrow{\{username,\ pw\}}$ server

user $\xleftarrow{ACK:OK \text{ or } ACK:NO!}$ server

CHAP Diagram

unused, random

user $\xleftarrow{authenticate,\ challenge}$ server

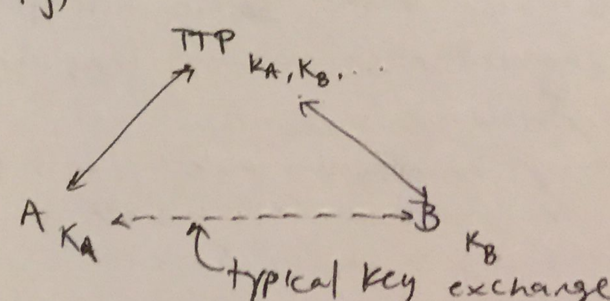
user $\xrightarrow{username,\ h(chal,\ pw)}$ server

checks hash → server (needs pw db access)

user $\xleftarrow{ACK:OK \text{ or } ACK:NO}$ server

Needham-Schroeder KX

- establishes session keys, instead of managing n^2 keys for n entities in a system (total)
- leverage a TTP (trusted third party)



Needham Schroeder Diagram + fix for replayability of MSG3

ALICE $\xrightarrow{\text{Alice} \| \text{Bob} \| R_1}$ CATHY (TTP)

ALICE $\xleftarrow{\{\text{Alice} \| \text{Bob} \| R_1 \| K_S \{\text{Alice} \| K_S\}_{K_B}\}_{K_A}}$ CATHY

- Encrypted using key that only Alice, Cathy know (K_A)
- Has R_1 from first message.

ALICE $\xrightarrow{\{\text{Alice} \| K_S\}_{K_B}}$ BOB

- Only Bob can decrypt as it is enc'd with K_B
- Now, any messages that have that K_s are known to be from Bob.

ALICE $\xleftarrow{\{R_2\}_{K_S}}$ BOB

- check that Alice is not Eve - if alice has shared session key, then she can respond with correct $R_2 - 1$

ALICE $\xrightarrow{\{R_2 - 1\} K_S}$ Bob

- Ensures to Bob that this is not Eve as she would not be able to decrypt R2.

Nonces
- R_1, R_2 above
- Not repeated (i.e. rand int, time...)

Alice $\xrightarrow{\text{Alice}}$ Bob

Alice $\xleftarrow{\{A, R_3\}_{K_S}}$ Bob

Alice $\xrightarrow{\text{Alice} \| \text{Bob} \| R_1 \| \{A, R_3\}_{K_S}}$ Cathy

Alice $\xleftarrow{\{A \| B \| R_1 \| K_S \{A \| K_S \| R_3\}_{K_B}\}_{K_A}}$ CATHY

Alice $\xrightarrow{\{\text{Alice} \| K_S \| R_3\}_{K_B}}$ Bob

Alice $\xleftarrow{\{R_2\}_{K_S}}$ Bob

Alice $\xrightarrow{\{R_2\} K_S}$ Bob

$\{R_2 - 1\}_{K_S}$