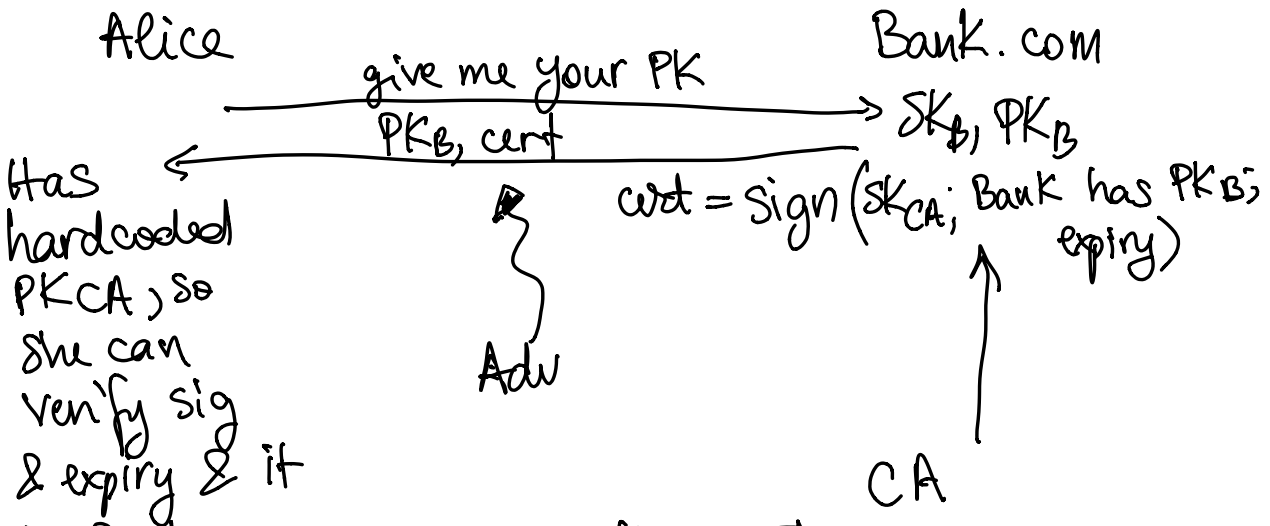
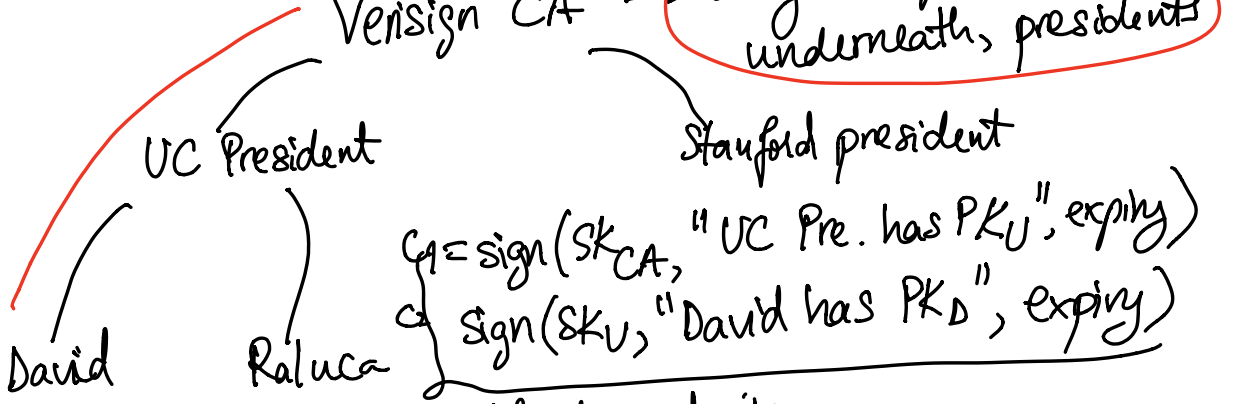


Better than TD:
+ can contact Bank (or anyone) to obtain PK

Certificate hierarchies & chains

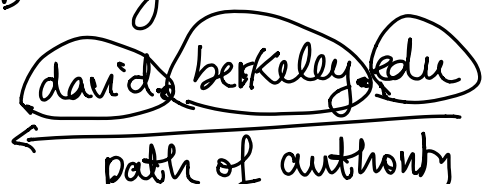
+ better scalability compared to TD

Verisign CA → only certifies level underneath, presidents



When I ask for David's PK; I will receive PK_D, C_1, C_2, PK_U

- check PK_U using C_1 using PK_{CA}
- check PK_D using C_2 and knowledge of PK_U



root servers serve cert. edu

Revocation

How can we revoke a certificate that has not yet expired?

- wait till expiry, make expiry shorter
- revocation lists: CA could push revocation Sign(SK_{CA}, "Revoke cert") into browsers; not ideal solution because browsers might not be downloading lists frequently

Long-term problem with CAs:
CAs can be compromised or could be deceived to sign incorrect certificates

- transparency logs promise to address this problem

Password hashing

CS 161: Computer Security

Prof. Raluca Ada Popa

Feb 28, 2020

Passwords

Tension between usability and security

choose memorable passwords

choose random and long passwords (hard to guess)

Attack mechanisms

- Online guessing attacks
  - Attacker tries to login by trying different user passwords in the live system
- Social engineering and phishing
  - Attacker fools user into revealing password
- Eavesdropping
  - Network attacker intercepts plaintext password on the connection
- Client-side malware
  - Key-logger/malware captures password when inserted and sends to attacker
- Server compromise
  - Attacker compromises server, reads storage and learns passwords

Defences/mitigations

Network eavesdropper:

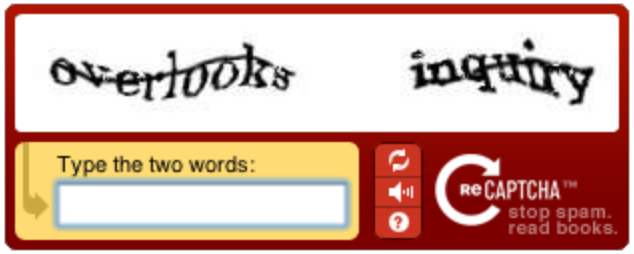
- Encrypt traffic using TLS (will discuss later)

Client-side malware: hard to defend

- Intrusion detection mechanisms – detect malware when it is being inserted into the network
- Various security software (e.g., anti-virus)
- Use two-factor authentication

Mitigations for online-guessing attacks

- Rate-limiting
  - Impose limit on number of passwords attempts
- CAPTCHAs: to prevent automated password guessing



- Password requirements: length, capital letters, characters, etc.

Mitigations for server compromise

- Suppose attacker steals the database at the server including all password information
- Storing passwords in plaintext makes them easy to steal
- Further problem: users reuse passwords at different sites!

Don’t store passwords in plaintext at server!

Hashing passwords

- Server stores hash(password) for each user using a cryptographic hash function
  - hash is a one-way function

username	hash of password
Alice	hash(Alice's password)
Bob	hash(Bob's password)

- When Alice logs in with password w (and provides w to server), server computes hash(w) and compares to Alice's record

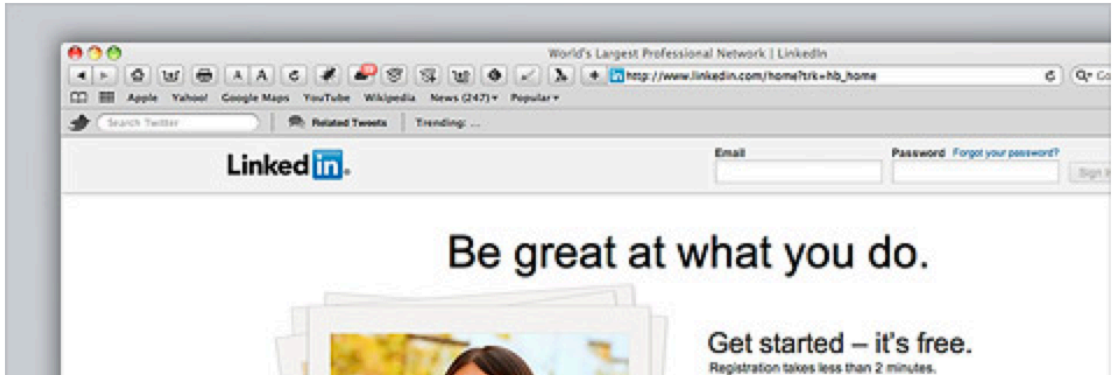
Password hashing: problems

- Offline password guessing
  - Dictionary attack: attacker tries all passwords against each hash(w)
  - If D is dictionary size, n number of hashes passwords, attack takes Dn
  - Study shows that a dictionary of 2^{20} passwords can guess 50% of passwords
- Amortized password hashing
  - Idea: One brute force scan for all/many hashes (D+n time)
  - Build table (H(password), password) for all 2^{20} passwords
  - Crack 50% of passwords in this one pass

THE CYBERCRIME ECONOMY

More than 6 million LinkedIn passwords stolen

By David Goldman @CNNMoneyTech June 7, 2012: 9:34 AM ET



LinkedIn was storing h(password)

"Link" was the number one hacked password, according to Rapid7. But many other LinkedIn users also picked passwords — "work" and "job" for example — that were associated with the career site's content.

Religion was also a popular password topic — "god," "angel" and "jesus" also made the top 15. Number sequences such as "1234" and "12345" also made the list.

Prevent amortized guessing attack

- Randomize hashes with salt
- Server stores (salt, hash(password, salt)), salt is random
- Two equal passwords have different hashes now
- Dictionary attack still possible, BUT need to do one brute force attack per hash now, not one brute force attack for many hashes at once
- Attacks takes Dn time instead of D+n time

Salted hash example

username	salt	hash of password
Alice	235545235	hash(Alice's password, 235545235)
Bob	678632523	hash(Bob's password, 678632523)

Attacker tries to guess Alice's password:

Computes table

'aaaaaa'	hash('aaaaaa', 235545235),
'aaaaab'	hash('aaaaab', 235545235),
…	
'zzzzzzz'	hash('zzzzzz', 235545235)

This table is useless for Bob's password because of different salt

Increase security further

- Would like to slow down attacker in doing a dictionary attack
- Use slow hashes = takes a while to compute the hash
- Define

  H(x) = hash(hash(hash(…hash(x))))

  use with x = password || salt

- Tension: time for user to authenticate & login vs attacker time
- If H is 1000 times slower and attack takes a day with H, attack now takes 3 years with F

Conclusions

- Do not store passwords in cleartext
- Store them hashed with salts, slower hash functions better