User Authentication (ctd.)

Security 1 (CM0475, CM0493) 2019-20
Università Ca' Foscari Venezia

Riccardo Focardi
www.unive.it/data/persone/5590470
secgroup.dais.unive.it

Università
Ca'Foscari
Venezia

Passwords

(summary of previous class)

Class: Something known

Passwords are stored “encrypted”:

- **One-way hash** of password and a **random *salt***
- **Iterated hash**: slow down brute force

Rainbow tables are a particularly efficient time-space tradeoff

- Countermeasure: random salt

Attacks on passwords (1)

Offline attacks: through rainbow tables and dictionaries, e.g. the 32M passwords leaked in the rockyou.com SQLi attack of 2009

Countermeasures:

- Salt, slow hashes, password policies (increase **entropy**)
- Restrict access to password file (mitigation)

Online attacks on single accounts: try easy passwords for one account

Countermeasures:

- Lock the account after MAX attempts (e.g. MAX=5)
- Same countermeasure for SIM, smartphone, bank PINs

Attacks on passwords (2)

Popular passwords: try a popular password on many accounts.

Countermeasures:

- Password policies
- Checking IPs (blacklisting)

Password guessing on a single user: guess an easy password for a specific user, e.g., using specific information about that user (name, age, etc.)

Countermeasures:

- User training
- Password policies

Attacks on passwords (3)

Workstation hijacking: wait until a workstation il left unattended (bypass user authentication)

Countermeasures:

- lock after a period on inactivity
- … same for smartphones

User mistakes: written, shared, and preconfigured passwords plus social engineering

Countermeasures:

- User training
- Use of two-factor authentication

Attacks on passwords (4)

Multiple password use: leaking a password that are reused across accounts is more impactful

Countermeasures:

- User training
- Forbid password reuse when possible (e.g. on devices in the same network)

Interception: password should only be transmitted is a secure way

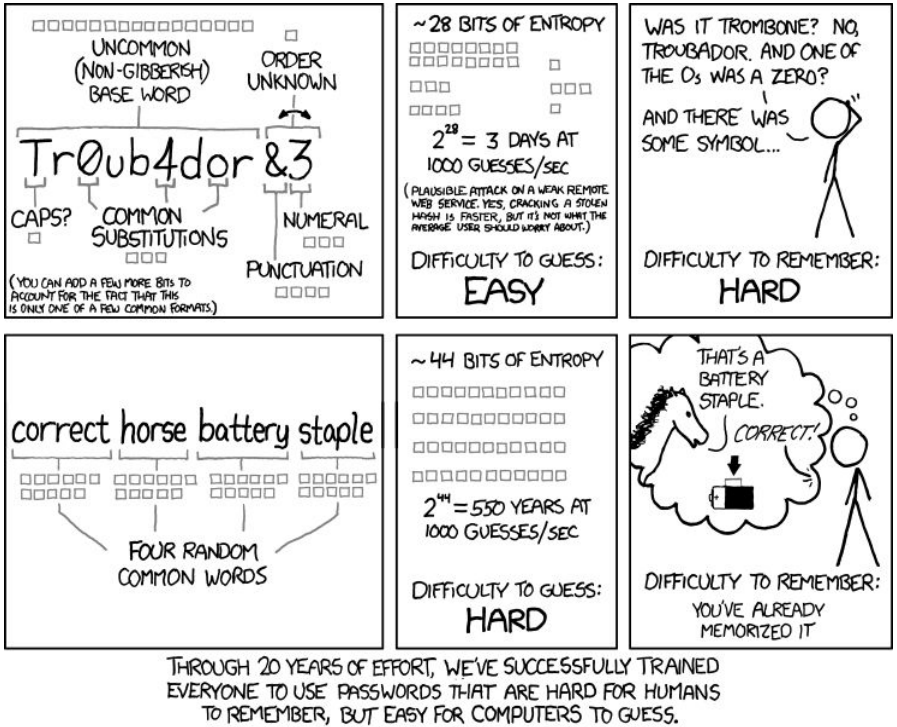
Countermeasures:

- Use a protocol that ensures authenticity of recipients and confidentiality (e.g. TLS)
- …. Just sending the password encrypted does not work!

Password policies

NIST SP 800-63-2 suggests the following alternative rules:

- Password must have at least sixteen characters (**basic16**)
- Password must have at least eight characters including an uppercase and lowercase letter, a symbol, and a digit. It may not contain a dictionary word (**comprehensive8**)



xkcd.com

Diceware

Passphrase of N words picked at random from a fixed list, by rolling 5 dice

- 5 dice gives $6^5 = 7776$ possible words
- Entropy for each word is $\log_2 7776 \sim 12.9$ bits

The **whole entropy** is thus 12.9 N

- for N=4 entropy is ~**52** bits
- for N=5 entropy is ~**64** bits
- for N=6 entropy is ~**77** bits

Word list: http://world.std.com/~reinhold/dicewarewordlist.pdf

Token-based authentication

Something possessed. Check the **possession** of a device

- ATM cards, credit cards, smartcards, One Time Password (OTP) generators, USB crypto-tokens

Memory cards

Passive card with a memory

Examples:

- Old ATM cards with magnetic stripe
- Hotel cards to open doors

When **paired with a PIN** the attacker needs to steal/duplicate both

Problems:

- Passive cards are usually simple to clone

Example:

- Old ATM cards were cloned by putting a fake reader and a camera (to also steal the PIN)

Smart cards

Smart token with embedded chip

Various devices:

- Standard smartcard
- USB token
- Small portable objects
- Bigger objects with display and/or keyboard

Smart card interface and protocol

Interface:

- **Contact**: a conductive contact plate on the surface of the card (typically gold plated) for transmission of commands, data, and card status
- **Contactless**: Both the reader and the card have an antenna, and communicate using radio frequencies

Protocol:

1. **Static**: token provides a fixed secret (as for passive cards)
2. **One time password** (OTP): the token generates a fresh OTP that is used for authentication
3. **Challenge-response**: a challenge is processed by the token that produces a response (e.g. digitally signed)

One Time Passwords (OTP)

Once a secret is leaked it can be used to authenticate many times:

- sniffed password
- cracked password hash
- cloned passive token

One Time Passwords (OTPs) are never reused

They mitigate password leakage/crack by allowing for a single authentication (es. bank OTPs)

⇒ The token and the computer system must be kept **synchronized** so the computer knows the password that is current for this token.

Lamport’s hash-based OTP

Given a secret s and a one-way hash function h we compute:

$$\mathbf{h^t(s)}, \text{ i.e., } h(h(\ldots\ h(s)\ldots\ ))\ t \text{ times}$$

We let the Claimant and the Verifier <u>share this value</u>

- The Claimant uses the list of passwords:
  $h^{t-1}(s),\ h^{t-2}(s),\ \ldots\ h(s),\ s$
- The Verifier computes $h(pwd)$ and checks if it is equal to the stored hash:
  $h(h^{t-1}(s)) == h^t(s)$
- If the check succeeds the Verifier stores $h^{t-1}(s)$

Lamport’s hash-based OTP

passwords: $h^{t-1}(s)$ $h^{t-2}(s)$ … $h(s)$ s

stored hashes: $h^{t}(s)$ $h^{t-1}(s)$ … $h^{2}(s)$ $h(s)$

Limitation: Only t authentications are possible

Security: Computing next passwords from the current is equivalent to compute the preimage of h, which is infeasible (h is one-way)

⇒ More secure than storing a shared secret “seed” used to generate the OTP

Case study: RSA seed breach

RSA SecurID Breach (March 2011)

- The values of secret “seeds” were stored insecurely and have been leaked through phishing

⇒ 40M of devices replaced, big companies attacked, huge image damage for RSA

Biometrics

Something inherent. Check **biometric** features of users

- Signatures, fingerprints, voice, face, hand geometry, retinal patterns, iris, ...

Biometrics

1. **Enrollment**: features are extracted and stored in database
2. **Verification**: features are extracted and compared with the stored ones

A delicate balance:

No impersonation (false positives) but correct user should be identified most of the times (no false negative)

Problems:

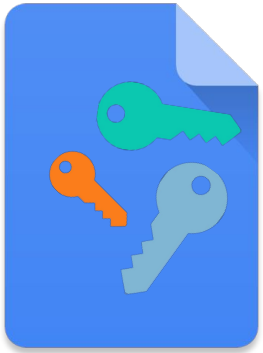
A breach in the biometric database has **high impact**:

- biometric data is unique, belongs to users
- differently from passwords it cannot be changed if leaked

New attacks: *adversarial machine learning*

Case study: Java keystores

Key Storage



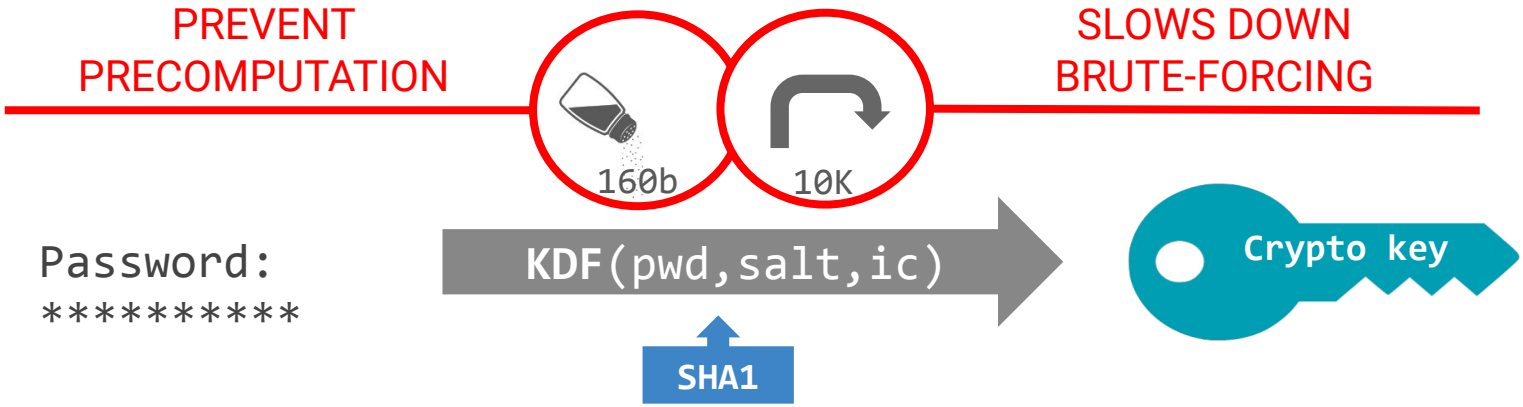
Key Confidentiality ✓
Key Integrity ✓
System Integrity ✓

******

Keystore

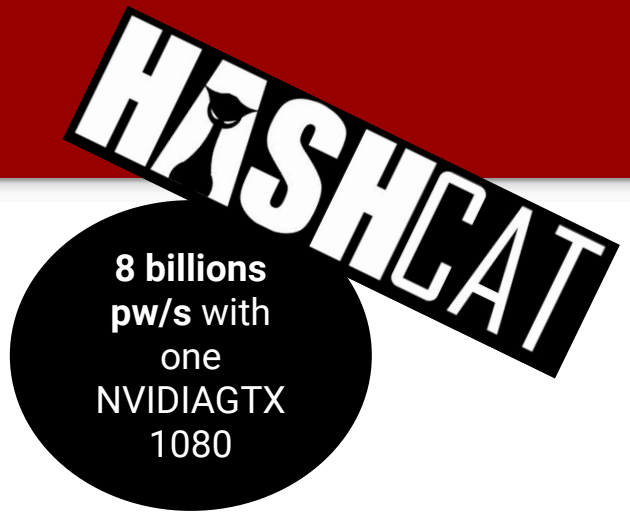
- File containing keys and certificates
- Password-protected

Key derivation function (KDF)



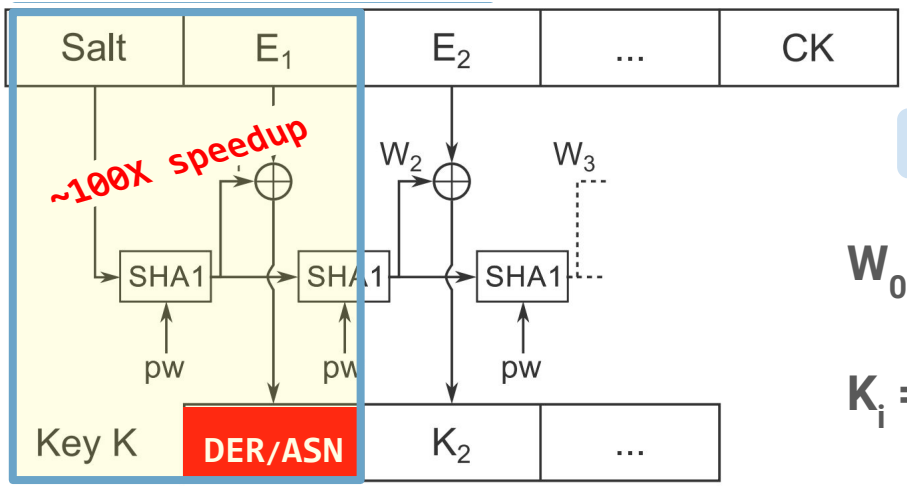
⇒ KDF is similar to password hashing but outputs a crypto key

Oracle JKS Password Cracking



Key Decryption in **JKS**

E = Encrypted Key



W = Keystream

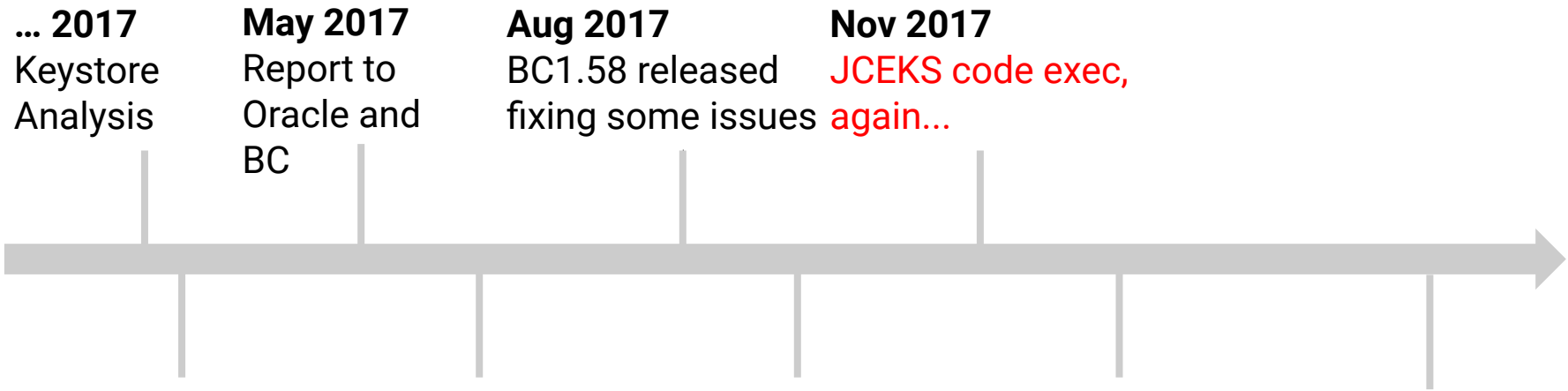
$W_0 = Salt$

$W_i = SHA1(pw||W_{i-1})$

$K_i = E_i \oplus W_i$

$CK = SHA1(pw||K)$

Java keystore vulnerabilities (NDSS18)



(For more information see the paper and the presentation at NDSS18)