PV204 Security technologies

Authentication: passwords, OTP, FIDO U2F

Petr Švenda ***svenda@fi.muni.cz*** ***@rngsec***
Centre for Research on Cryptography and Security, Masaryk University

CROCS
Centre for Research on Cryptography and Security

🙏 *Please report any inaccuracies or suggestions for improvements here:*
https://drive.google.com/file/d/1qspWBELzanOFIu2AYdtfZvwN0bOmnRv-/view?usp=sharing

Join at

slido.com

#pv204_2022

- **Place/upvote questions in slido while listening to lecture video**
- **We will together discuss these during every week lecture Q&A**

COURSE TRIVIA: PV204_00_COURSEOVERVIEW_2022.PDF

Basic terms

- Identification
  - Establish what the (previously unknown) entity is
- Authentication
  - Verify if entity is really what it claims to be
- Authorization (access control)
  - Define an access policy to use specified resource
  - Check if entity is allowed (authorized) to use resource
- Authentication may be required before an entity allowed to use resource to which is authorized

Options for authentication

- Something you:
  1. Know (password, key)
  2. Have (token, smartcard)
  3. Are (biometrics)
- Combination of multiple options – two-factor authentication (or more)

1. Registration phase (how is new user added)
2. Verification phase (how is user's claimed identity verified)
3. Recovery phase (what if user forgot/lost authentication credentials)

PASSWORDS

Mode of usage for passwords

- Verify by direct match (provided_password == expected_password?)
  - Example: *HTTP basic access* authentication
  - Be aware of plaintext storage on server
  - Be aware of potential side-channels (mismatch on Xth character)
- Verify by match of derived value (hash(password | salt))
  - Be aware of rainbow tables and brute-force crackers
- Derive key: Password → cryptographic key
  - Example: key = PBKDF2(password)
- Used to establish authenticated key
  - Example: Password + Diffie-Hellman → authenticated key…

Problems associated with passwords

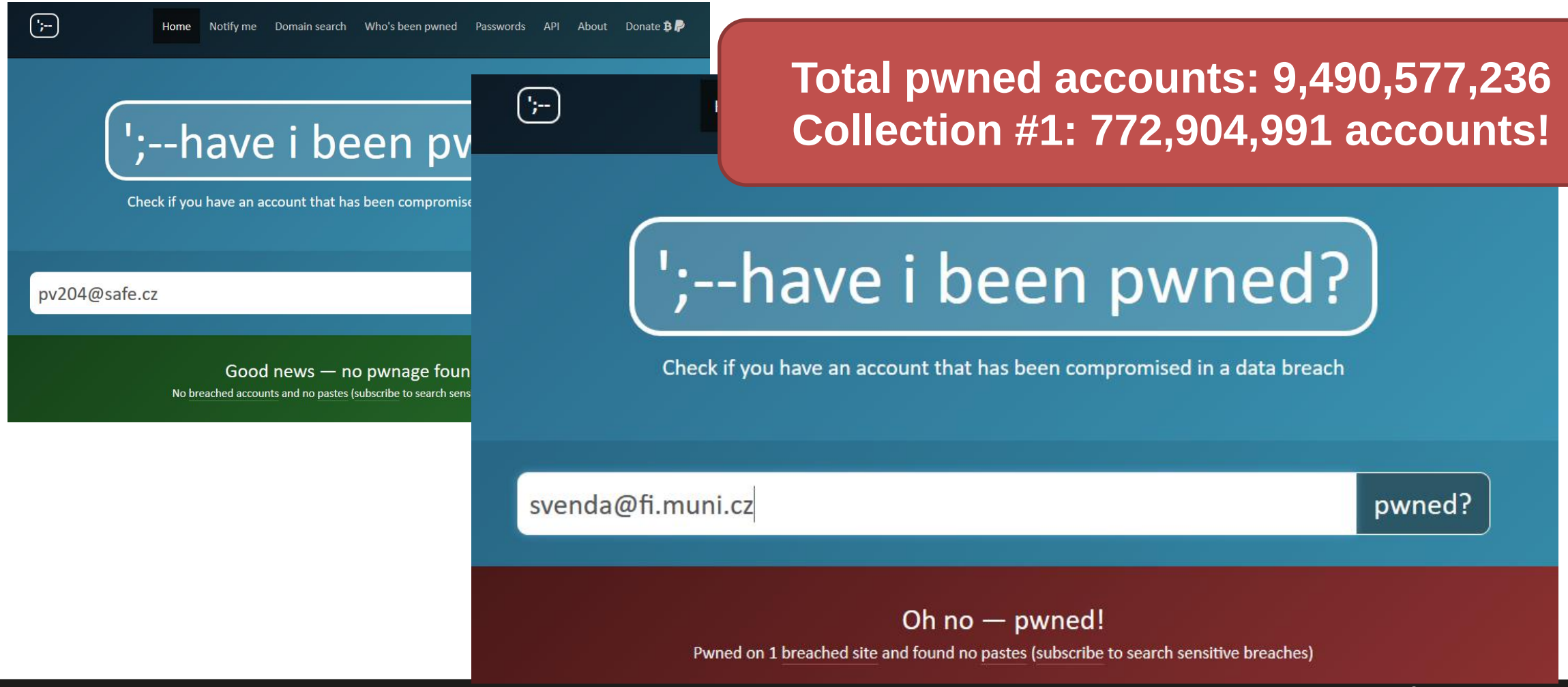
- How to create strong password?
- How to use password securely?
- How to store password securely?
- Same value is used for the long time (exposure)
- Value of password is independent from the target operation (e.g., authorization of bank transfer request)
- …

Where the passwords can be compromised?

1. Client side (malware on user computer)
2. Database storage
   - Cleartext storage
   - Backup data (“tapes”)
   - Server compromise, misconfiguration
3. Host machine (memory, history, cache)
4. Network transmission (network sniffer, proxy logs)
5. Hardcoded secrets (inside app binary)

- Difficult to detect compromise and change after the exposure

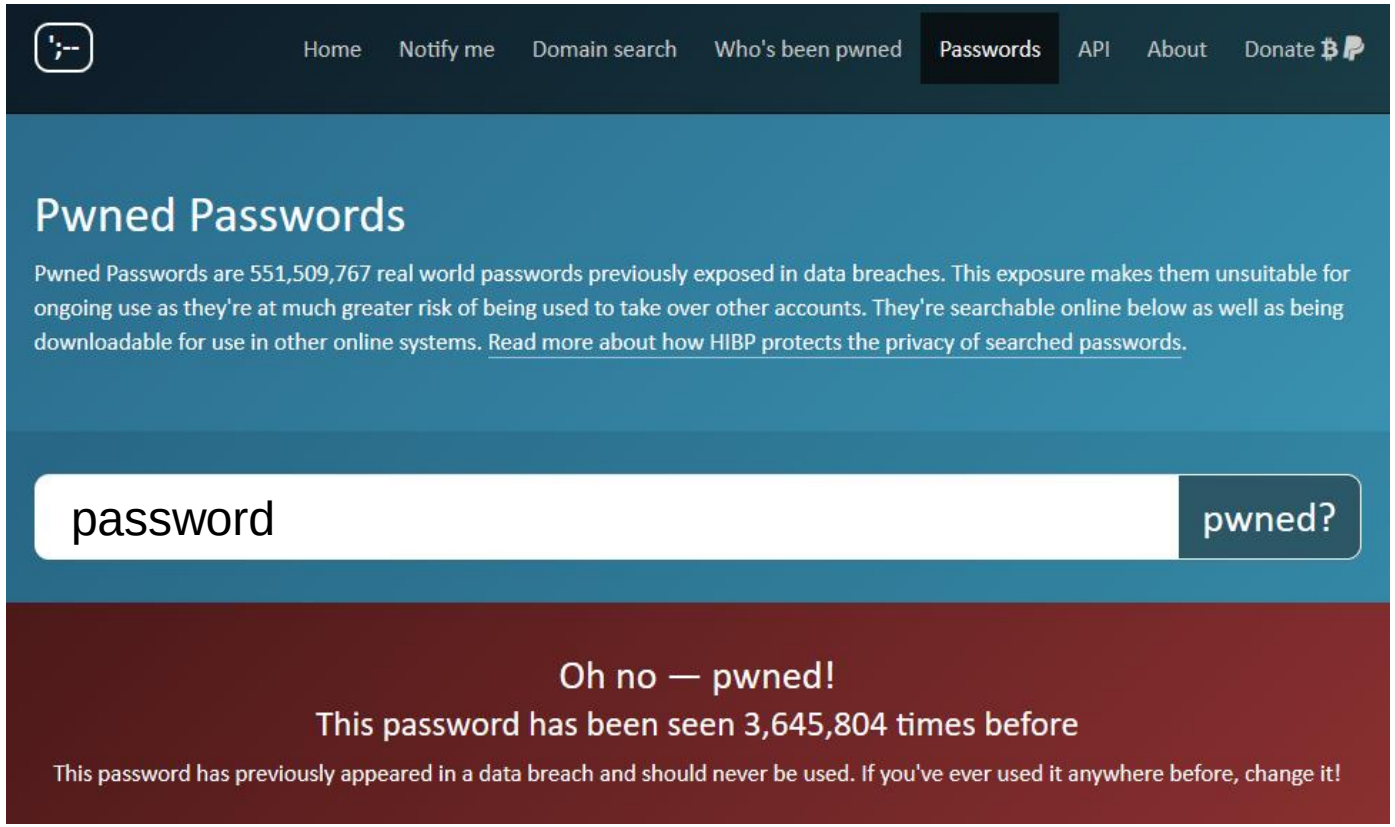
https://haveibeenpwned.com/ (Troy Hunt)



Total pwned accounts: 9,490,577,236
Collection #1: 772,904,991 accounts!

https://haveibeenpwned.com/Passwords

- Check how many times was given password found in leaked datasets



Password “hardening” ideas

1. Hash password by one-way function (shall be hard to invert)
2. Slowdown cracking attempts (less potential passwords tried)
3. Enable users to have long, random and unique passwords
4. Have unique password for every authentication attempt
5. Replace/complement passwords with something else (e.g., smartcard)
6. Bind response to server domain name (to prevent phishing)

In follow-up slides, we will discuss these ideas one by one

IDEA: HASH PASSWORDS

/etc/{passwd,shadow}

Central file(s) describing UNIX user accounts.

/etc/passwd

- Username
- UID
- Default GID
- GCOS
- Home directory
- Login shell

/etc/shadow

- Username
- Encrypted password
- Date of last pw change.
- Days 'til change allowed.
- Days `til change required.
- Expiration warning time.
- Expiration date.

student:x:1000:1000:Example User,,555-1212,:/home/student:/bin/ba

student:1w/UuKtLF$otSSvXtSN/xJzUOGFElNz0:13226:0:99999

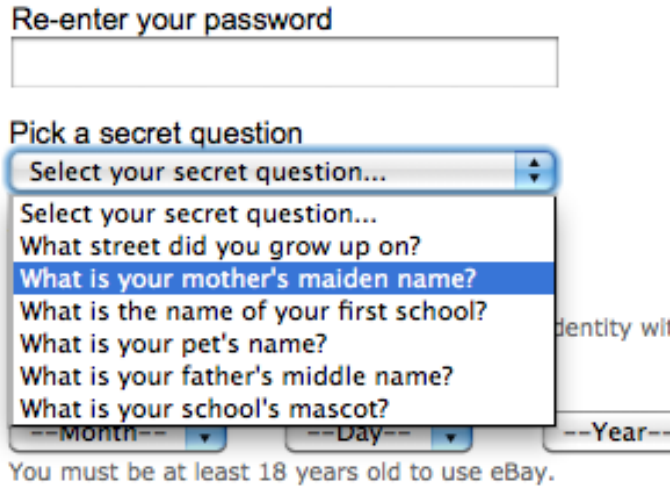
```
[root@arch01 ~]# cat /etc/shadow | sed 's/michael/test/' | sed 's/mbo/joe/'
root:$6$4GxAA08J$AB7vFkLSCxtVdVMcPav8jZ5u4ZsyG22hy1cqWPdnQgqL84VesJNQYFXSwhfwkhT
UeHNxYwjjUGe8U/sjITBhq/:16672:::::
bin:x:14871:::::
daemon:x:14871:::::
mail:x:14871:::::
ftp:x:14871:::::
http:x:14871:::::
uuidd:x:14871:::::
dbus:x:14871:::::
nobody:x:14871:::::
systemd-journal-gateway:x:14871:::::
systemd-timesync:x:14871:::::
systemd-network:x:14871:::::
systemd-bus-proxy:x:14871:::::
systemd-resolve:x:14871:::::
systemd-journal-upload:!!:16672:::::
systemd-journal-remote:!!:16672:::::
avahi:!:16672:::::
polkitd:!!:16672:0:99999:7:::
joe:$6$TA4PslzF$ch961z/ppk1VrmVAqSjSEDf75FIahttselx/bsDdjSXLt8cmsIoX9eAKfVm8epuD
KGvYV1xkohA37aeEvmu8d1:16672:0:99999:7:::
gdm:!!:16685:::::
test:$6$PNkLwU7L$2Hm8YRMGgRoxxt4srAzGBZJFxU7S
kZ1PwBzY1aHAvZu29wSBpJ0:16735:0:99999:7:::
```

Joe; insecure

(Hashed-)Password cracking

- Scenario: dump of database with password hashes, find original password
- Password cracking attacks
  - Brute-force attack (up to 8 characters)
  - Dictionary attack (inputs with higher probability tried first)
  - Patterns: Dictionary + brute-force (Password[0-9]*)
  - Rainbow tables (time-memory trade-off)
  - Parallelization (many parallel cores)
  - GPU/FPGA/ASIC speedup of cracking
- Tools
  - Generic: John the Ripper, Brutus, RainbowCrack…
  - Targeted to application: TrueCrack, Aircrack-NG…

Password reality (from many breaches + pwd cracking)



- User has usually weak password
  - >60% were (dictionary) brute-forced
- Server/service is frequently compromised
  - Server-side compromises are now very frequent
- Users do not use unique passwords
  - Gawker/root.com leak: 76% had the exact same password
- Different authentication channels may not be independent
  - Web-browsing + SMS on smart phones?
- Account recovery is often easier to guess than original password

Insecure password handling … what is the attack?

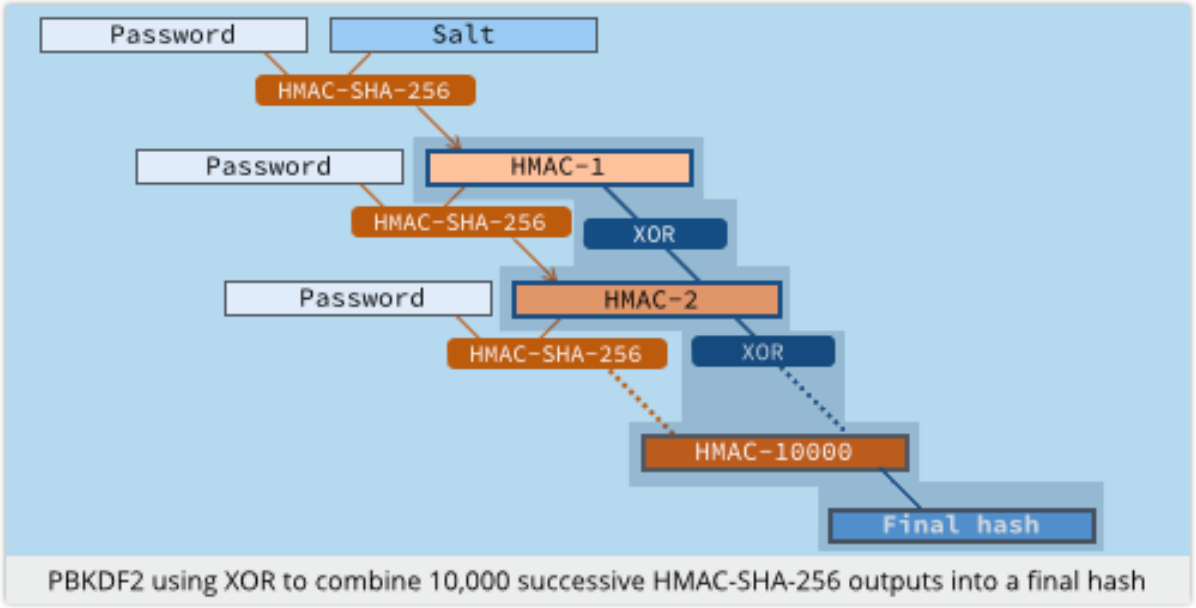
- Verify by direct match (provided_password == expected_password?)
  - Attack: compromise plain passwords on server
- $pwdTag_i$ = SHA-2(“password”)
  - Same passwords from multiple users => same resulting pwdTag
  - Attack: Large pre-computed “rainbow” tables allow for very quick check common passwords
- $pwdTag_i$ = SHA-2(“password” | salt)
  - Use of rainbow tables “prevented” by addition of random (and potentially public) *salt*
  - Attack: dictionary-based brute-force still possible
- $pwdTag_i$ = AES(“password”, secret_key)
  - Attack: If secret_key is leaked => direct decryption of all stored pwdTags => passwords

Some issues addressed by PAKE (Password Authenticated Key Exchange) protocols – future lecture

IDEA: SLOWDOWN CRACKING ATTEMPTS

Derivation of secrets from passwords

- PBKDF2 function, widely used
  - Password is key for HMAC
  - Salt added
  - Many iterations to slow derivation



PBKDF2 using XOR to combine 10,000 successive HMAC-SHA-256 outputs into a final hash

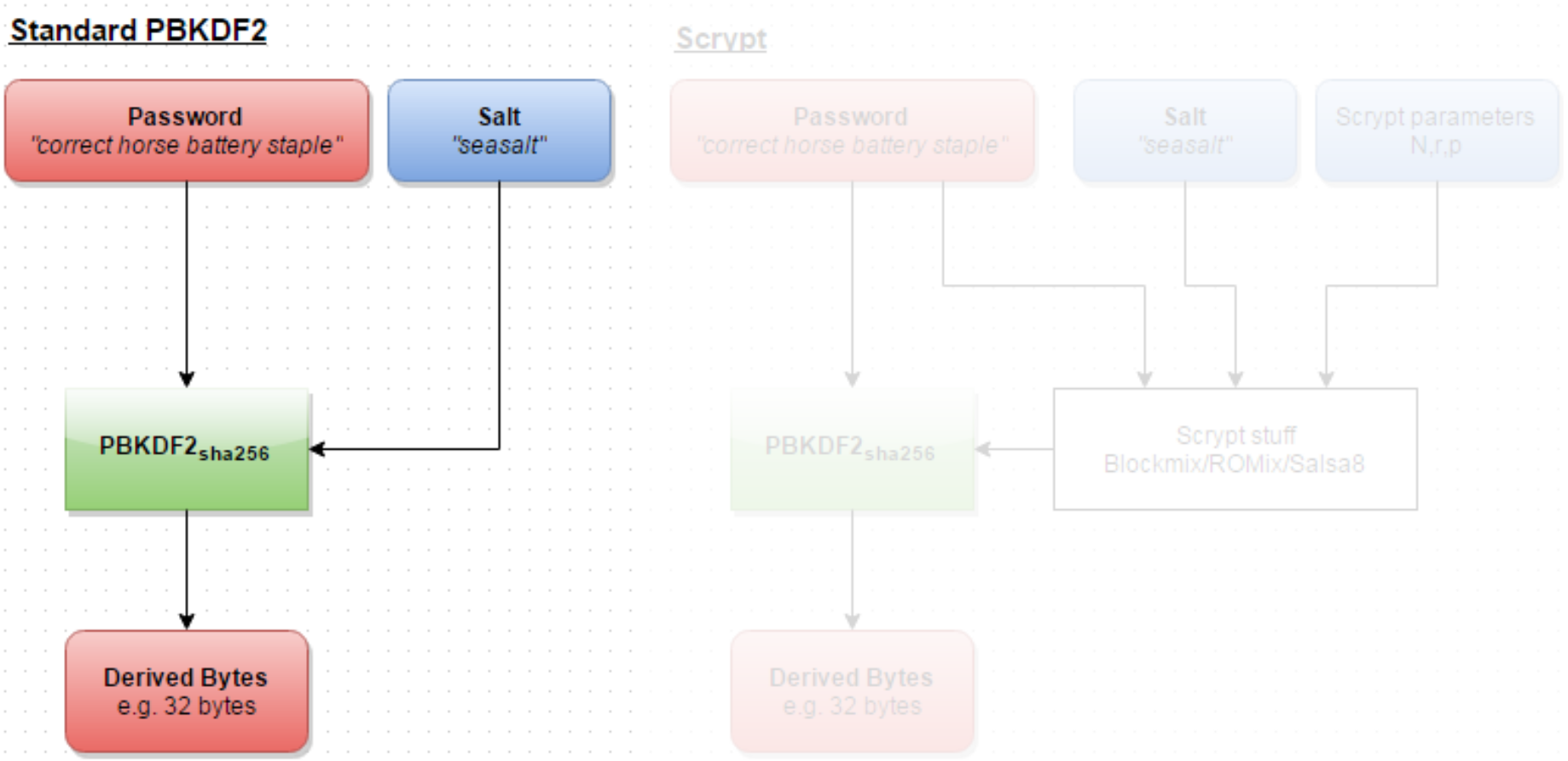
Source: https://nakedsecurity.sophos.com

- Problem with custom-build hardware (GPU, ASIC)
  - Repeated iterations not enough to prevent bruteforce
  - (or would be too slow on standard CPU – user experience)
- Solution: function which requires large amount of memory

scrypt – memory hard function

- Design as a protection against cracking hardware (usable against PBKDF2)
  - GPU, FPGA, ASICs…
  - https://github.com/wg/scrypt/blob/master/src/main/java/com/lambdaworks/crypto/SCrypt.java
- Memory-hard function
  - Force computation to hold *r* (parameter) blocks in memory
  - Uses PBKDF2 as outer interface
- Improved version: NeoScrypt (uses full Salsa20)

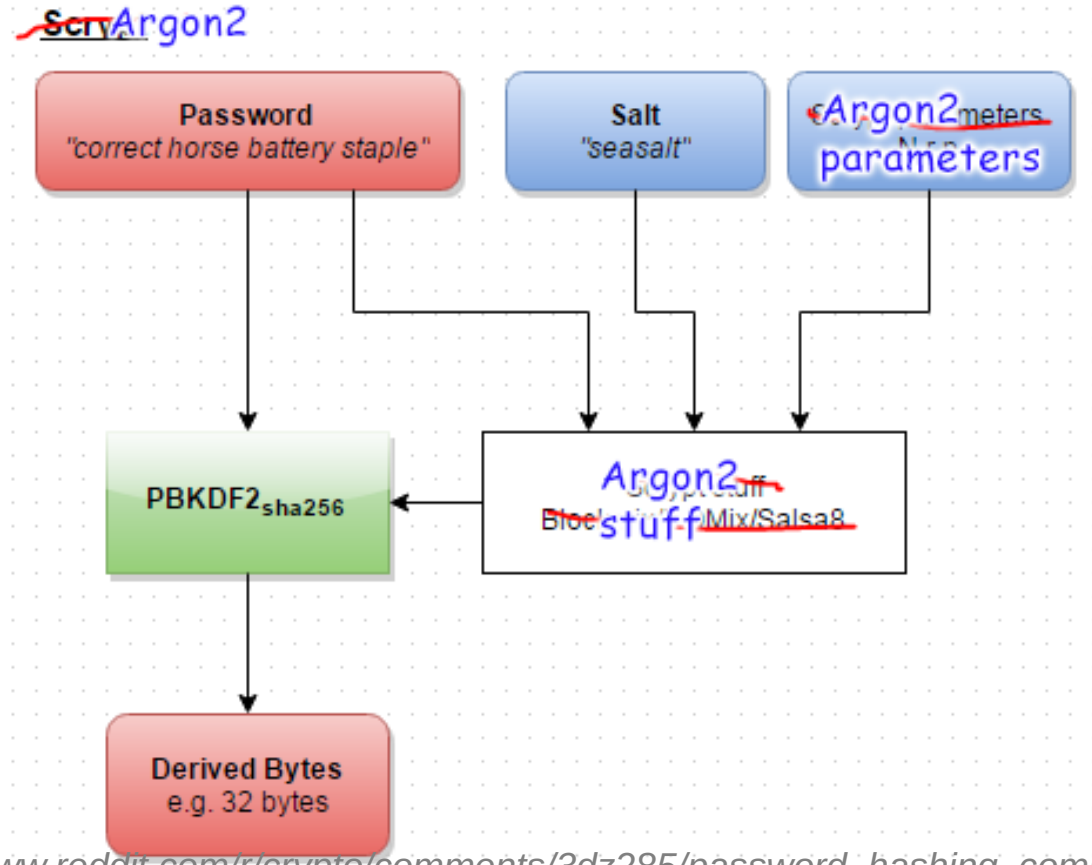
Reuse of external PBKDF2 structure



https://www.reddit.com/r/crypto/comments/3dz285/password_hashing_competition_phc_has_selected/

Argon2

- Password hashing competition (PHC) winner, 2013



https://www.reddit.com/r/crypto/comments/3dz285/password_hashing_competition_phc_has_selected/

Problem solved?

- *To*: cfrg at irtf.org
- *Subject*: [Cfrg] Argon2i, scrypt, balloon hashing, ...
- *From*: Phillip Rogaway <rogaway at cs.ucdavis.edu>
- *Date*: Mon, 15 Aug 2016 15:37:21 -0700 (Pacific Daylight Time)
- *Archived-at*: <https://mailarchive.ietf.org/arch/msg/cfrg/Xu9hCT6dqVmD50CezeR1MFsos0o>
- *Delivered-to*: cfrg at ietfa.amsl.com
- *In-reply-to*: <mailman.995.1471241877.1171.cfrg@irtf.org>
- *List-archive*: <https://mailarchive.ietf.org/arch/browse/cfrg/>
- *List-help*: <mailto:cfrg-request@irtf.org?subject=help>
- *List-id*: Crypto Forum Research Group <cfrg.irtf.org>
- *List-post*: <mailto:cfrg@irtf.org>
- *List-subscribe*: <https://www.irtf.org/mailman/listinfo/cfrg>, <mailto:cfrg-request@irtf.org?subject=subscribe>
- *List-unsubscribe*: <https://www.irtf.org/mailman/options/cfrg>, <mailto:cfrg-request@irtf.org?subject=unsubscribe>
- *References*: <mailman.995.1471241877.1171.cfrg@irtf.org>
- *User-agent*: Alpine 2.00 (WNT 1167 2008-08-23)

```
I would like to gently suggest the CFRG not move forward with blessing any memory-hard hash function
at this time. The area seems too much in flux, at this time, for this to be desirable. Really nice
results are coming out apace. Standards can come too early, you know, just as they can come out too
late.


phil
```

Problem: situation with PHC winner still unclear in 2019 ☹

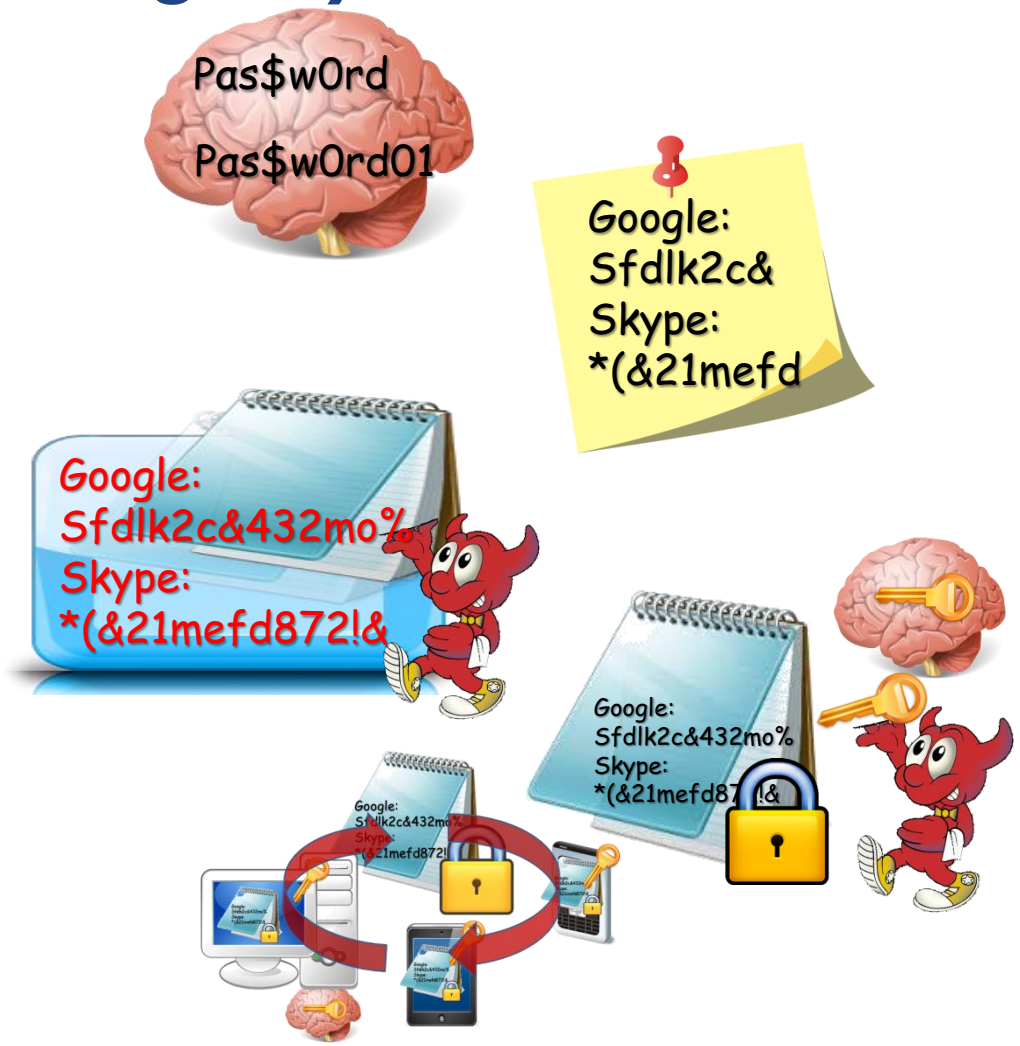
https://www.ietf.org/mail-archive/web/cfrg/current/msg08439.html

IDEA: LONG, RANDOM AND UNIQUE PASSWORDS

PASSWORD MANAGERS

Evolution of password (managers)

1. Human memory only
2. Write it down on paper
3. Write it into file
4. Use local password manager



Remote password managers

KeePass+Dropbox
LastPass
1Password
MozillaSync
Firefox Lockwise
…

- Firefox Lockwise https://www.mozilla.org/en-US/firefox/lockwise/
  - Part of the standard Firefox installation, sync between devices
  - Automatically checks for password leakage (Firefox Monitor)

c|net Search CNET Reviews Ne

CNET › Security › LastPass CEO reveals details on security breach

LastPass CEO reveals details on security breach

CEO of the password management company, which is dealing with a likely breach, tells PC World that users with strong master passwords should be safe,

users with strong master passwords should be safe,

0 / / / / / more +

Following yesterday's **revelation of a likely security breach** at password management company LastPass, the company's CEO is revealing more details about the incident and trying to offer some comfort and advice to his users.

Speaking yesterday with **PC World**, LastPass CEO Joe Siegrist admits he may have been too "alarmist" in sounding the alarm bell over the potential security breach. But the anomalies the company found when looking over its logs raised too much of a red flag.

Siegrist explained that he doesn't think a lot of data would've been hacked, but just enough to capture a small number of user names and passwords

But passwords are encrypted, right?

Case study

PASSWORD MANAGER FOR MULTIPLE DEVICES

Functional and security assumptions

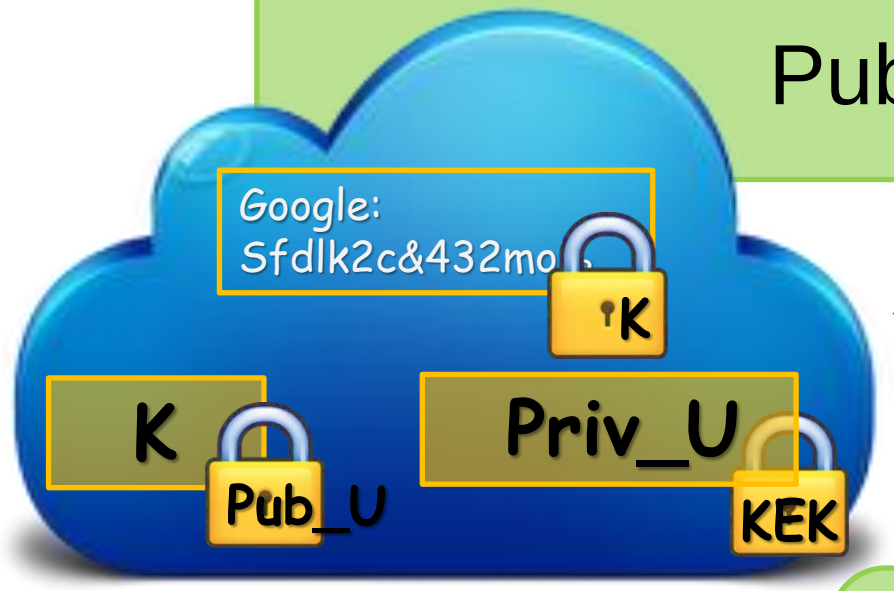
- Functional
  - User stores fixed secrets (passwords…)
  - User has multiple connected devices
  - Easy to use ☺
- Security
  - Service can’t be trusted
  - User chooses weak password
  - Devices can be lost (and later revoked)
  - User has independent channel (phone)

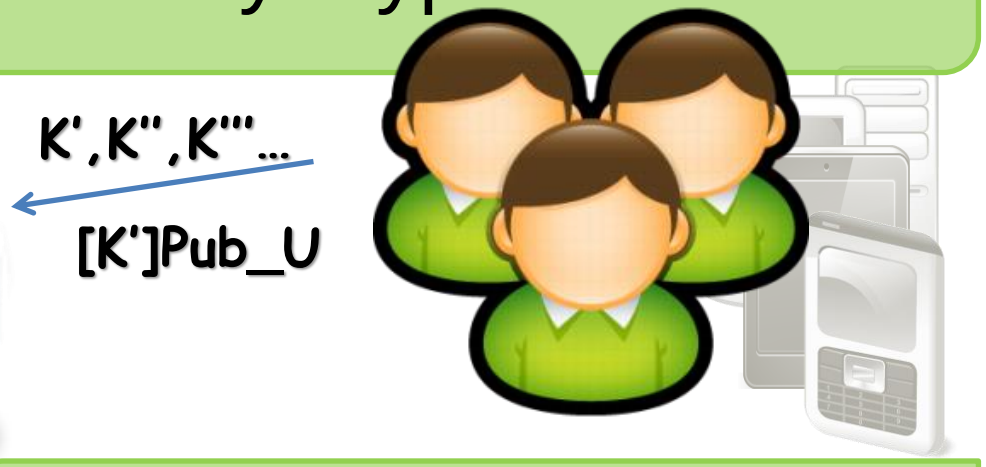
Main security design principles

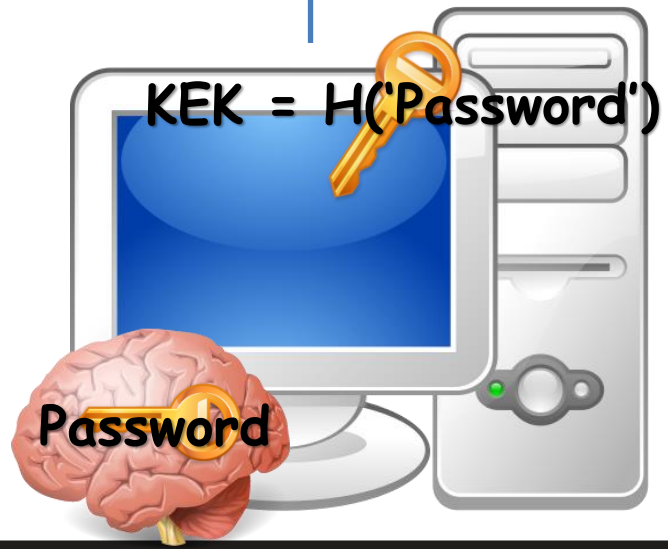
- Treat storage service as untrusted and perform security sensitive operations on client
- Make necessary trusted component as small as possible
- Prevent offline brute-force, but don’t expect strong password from user
  - add entropy from other source
- Make transmitted sensitive values short-lived
- Trusted hardware can provide additional support

Public-key cryptography indirection

Public-key crypto indirection



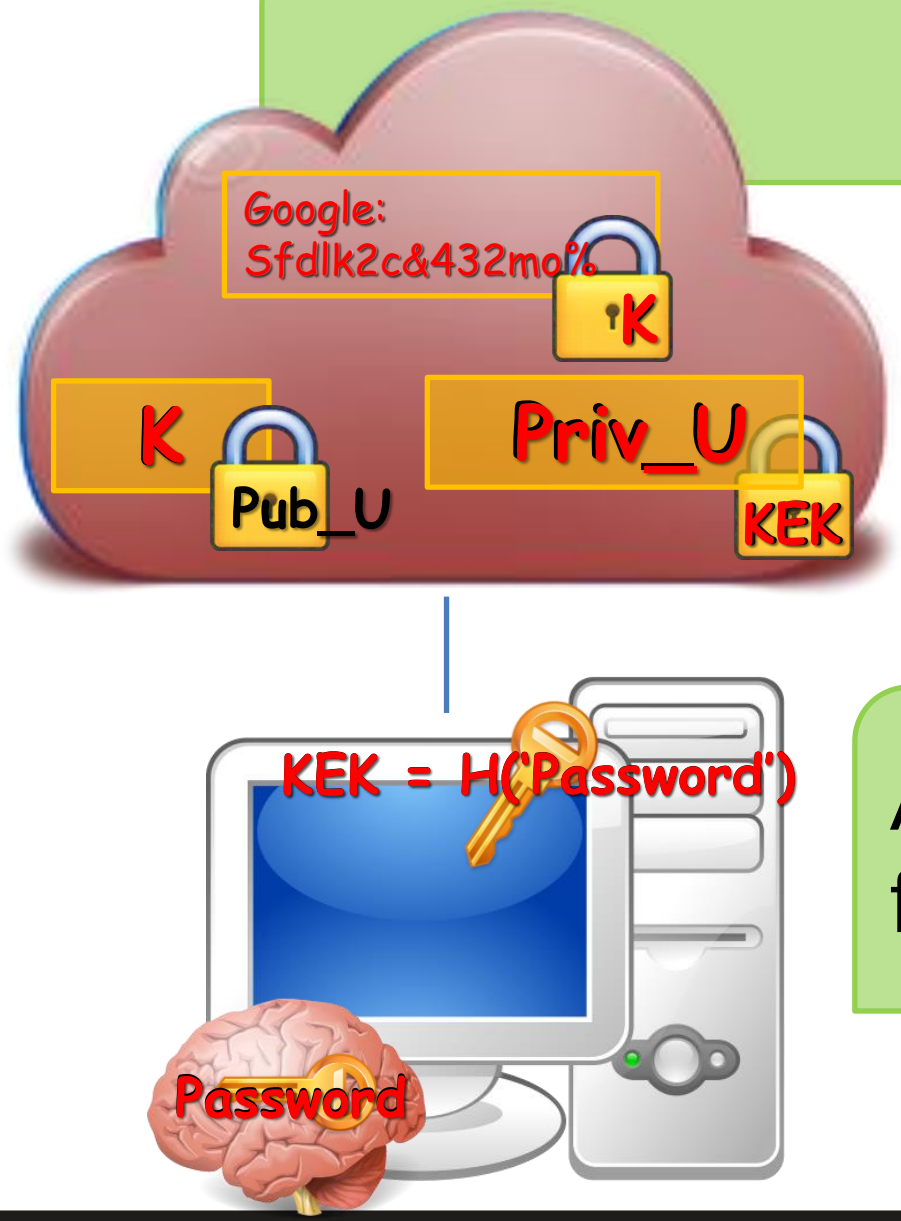




Public-key crypto indirection allows for asynchronous change of K

Long private key can be also stored on Service

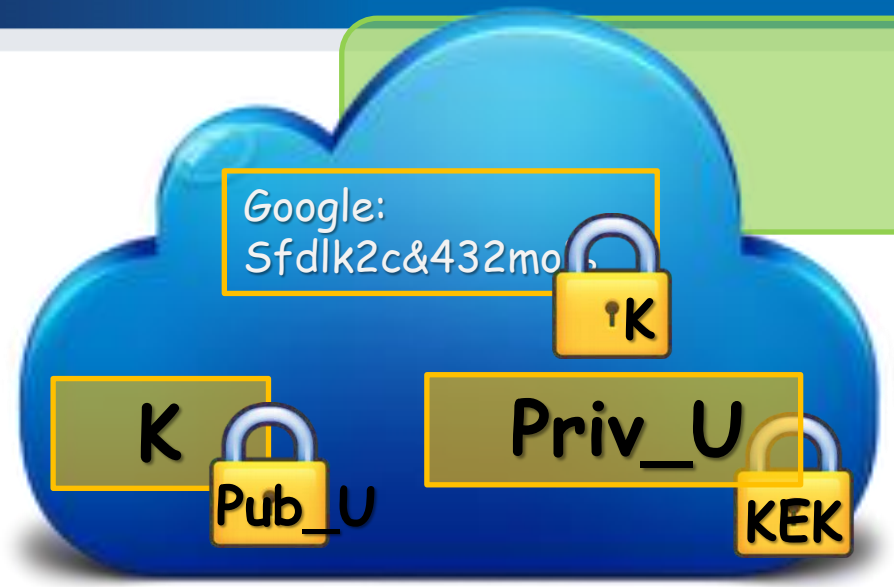
Weak password?



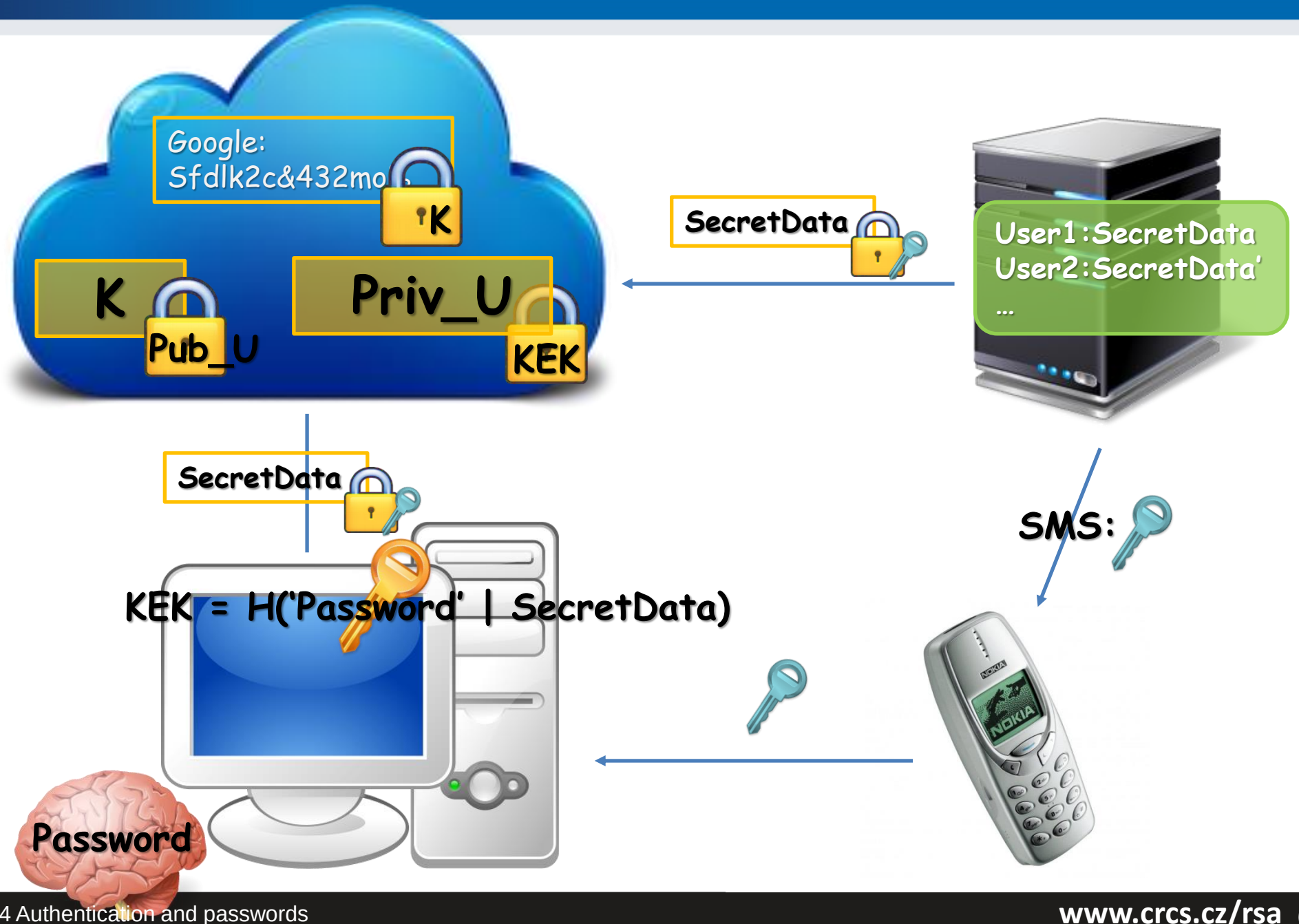
Users tend to have weak passwords…

Attacker has motivation for attacking the Service!

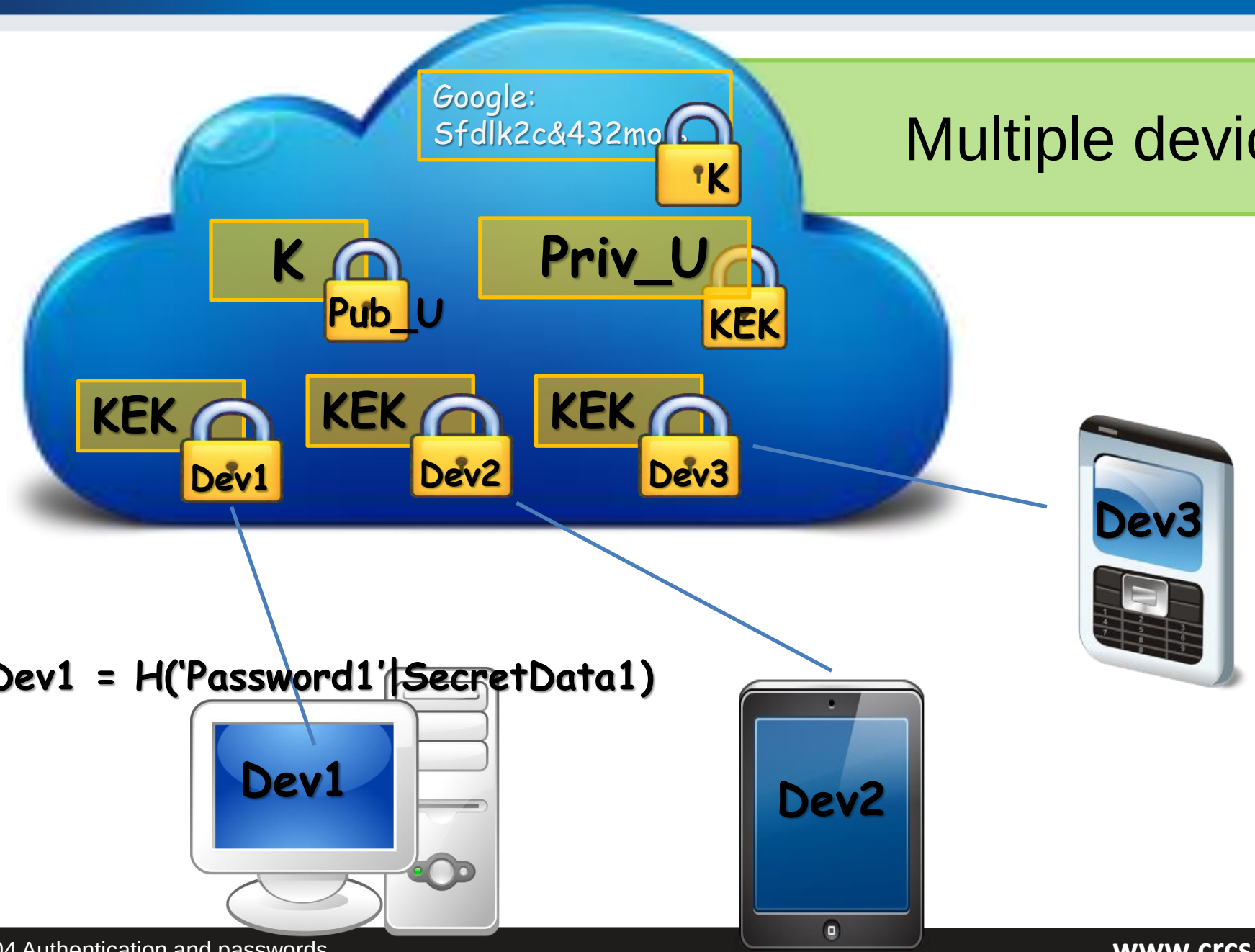
Trusted server/element



Separate trusted entities provide additional data


Google:
Sfdlk2c&432mo
K
K
Pub_U
Priv_U
KEK
SecretData
User1:SecretData
User2:SecretData'
...
SecretData
SMS:
KEK = H('Password' | SecretData)
Password

Multiple devices



Other operations

- Device management (new, remove, revoke)
- Device authentication
- Group management (users, boards, secrets)
- Password change, private key change
- Access recovery
- …

Devil is in the details…

Do we have some implementations?

- Apple's service showcased in 2013
- Lack of details until iOS Security report 02/2014
  - https://www.apple.com/business/docs/iOS_Security_Guide.pdf
- https://blog.cryptographyengineering.com/2016/08/13/is-apples-cloud-key-vault-crypto/ (M.Green)

Apple's iCloud Keychain

- Multiple similarities to the described example
  - Layer of indirection via asymmetric cryptography
  - Support for multiple devices
  - Asynchronous operations via application tickets
  - Authorization and signature of additional devices
  - User phone registered and required
- Still reliance on user's (potentially weak) password
  - But limited number of tries allowed
- Trusted component of iCloud realized via internal HSM
  - Recovery mode with 4-digit code (default, can be set longer)
  - HSM will decrypt recovery key only after code validation
  - Note: only 4 digits is not an issue here – HSM enforce limited # retries

IDEA: HAVE UNIQUE PASSWORD FOR EVERY AUTHENTICATION ATTEMPT

ONE-TIME PASSWORDS

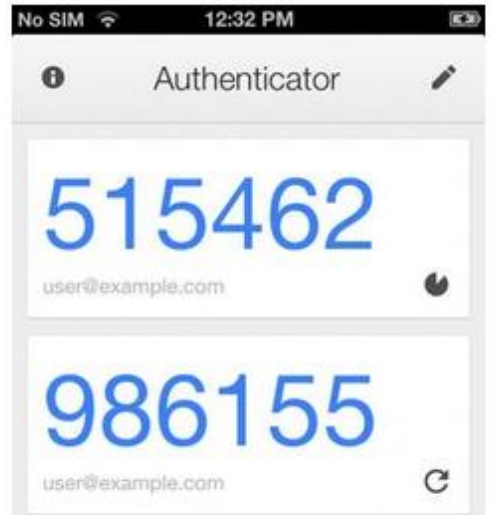
Recall: Problems associated with passwords

- How to create secure password?
- How to use password securely?
- How to store password securely?
- Same value is used for the long time (exposure)
- Value of password is independent from target operation (e.g., authorization of request)
- …

One-time passwords tries to address these issues

HMAC-based One-time Password Algorithm (RFC 4226)

- HMAC-based One-time Password Algorithm (HOTP)
  - Secret key K
  - Counter (challenge) C
  - $HMAC(K,C) = SHA1(K \oplus 0x5c5c\ldots \| SHA1(K \oplus 0x3636\ldots \| C))$
  - $HOTP(K,C) = Truncate(HMAC(K,C)) \& 0x7FFFFFFF$
  - 0x7FFFFFFF mask to drop most significant bit (portability)
  - $HOTP\text{-}Value = HOTP(K,C) \bmod 10^d$ (d … # of digits)
- Many practical implementations
  - E.g., Google Authenticator
- https://en.wikipedia.org/wiki/HOTP



HOTP – items, operations

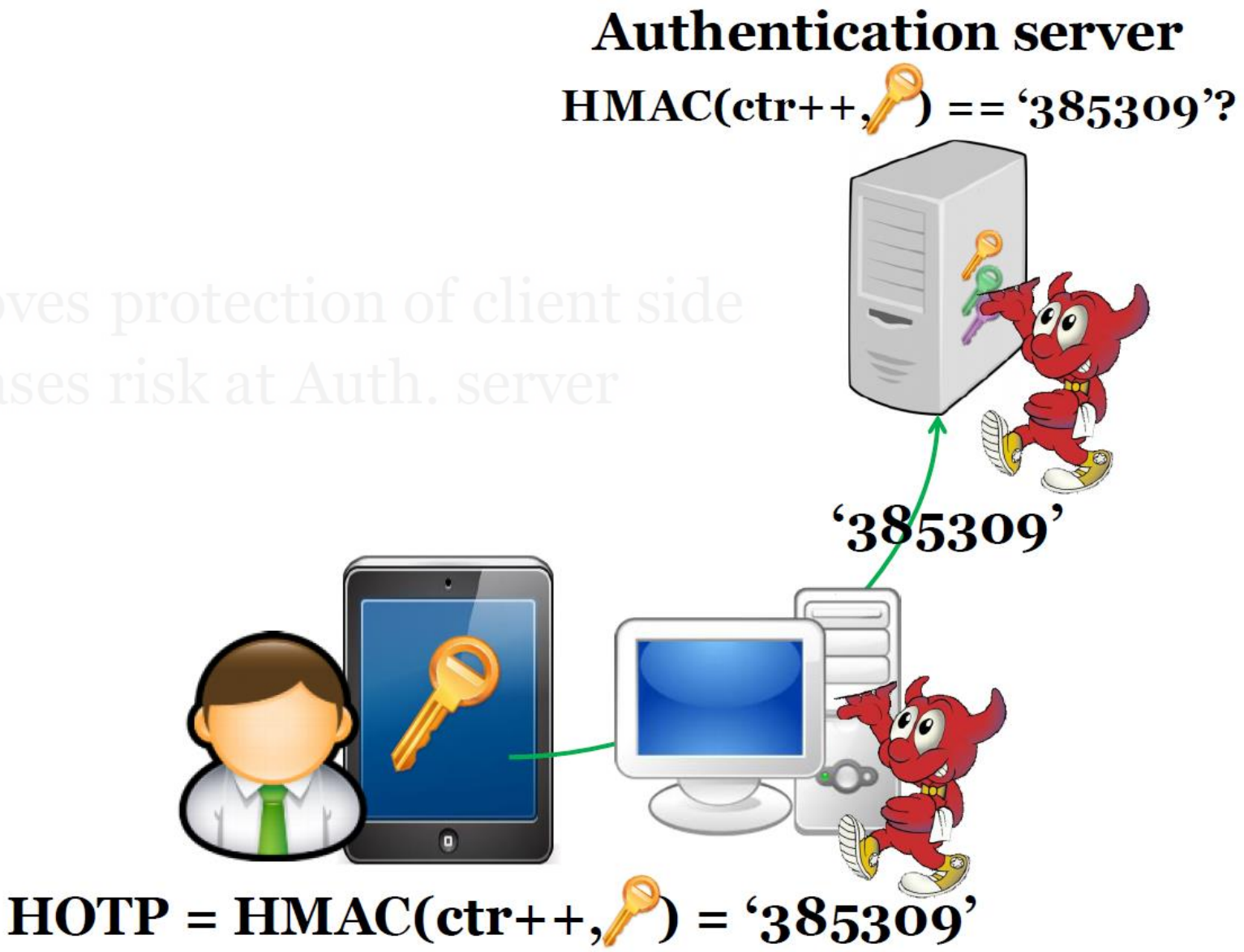
- Logical operations
  1. Generate initial state for new user and distribute key
  2. Generate HOTP code and update state (user)
  3. Verify HOTP code and update state (auth. server)
- Security considerations of HOTP
  - Client compromise
  - Server compromise
  - Repeat of counter/challenge
  - Counter mismatch tolerance window

Time-based One-time Password Algorithm

- Very similar to HOTP
  - Time used instead of counter
- Requires synchronized clocks
  - In practice realized as time window
- Tolerance to gradual desynchronization possible
  - Server keeps device's desynchronization offset
  - Updates with every successful login

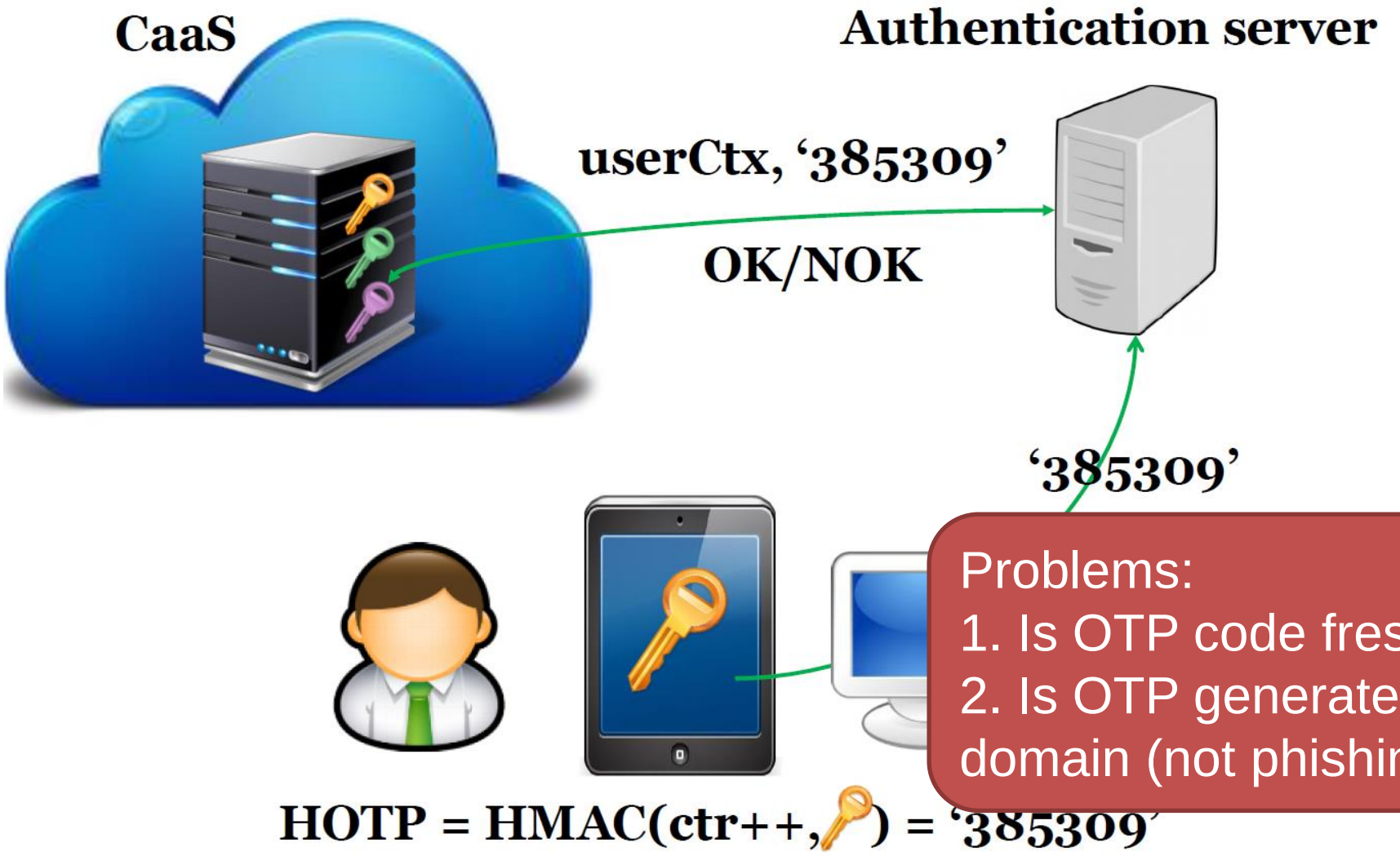
OCRA: OATH Challenge-Response Algorithm

- Initiative for Open Authentication (OATH)
- OCRA is authentication algorithm based on HOTP
- OCRA code = CryptoFunction(K, DataInput)
  - *K*: a shared secret key known to both parties
  - *DataInput*: concatenation of the various input data values
    - Counter, challenges, H(PIN/Passwd), session info, H(time)
  - Default *CryptoFunction* is HOTP-SHA1-6
  - https://tools.ietf.org/html/rfc6287
- Don’t confuse with Oauth (delegation of authentication)
  - The OAuth 2.0 Authorization Framework (RFC6749)
  - TLS-based security protocol for accessing HTTP service


Authentication server
HMAC(ctr++, 🔑) == '385309'?
'385309'
HOTP = HMAC(ctr++, 🔑) = '385309'

Increased risk at *OTP verification server

- More secure against client compromise
  - Using OTP instead of passwords, KDF(time|key),
- But what if server is compromised?
  - database hacks, temporal attacker presence
  - E.g., Heartbleed – dump of OTP keys
- Possible solution
  - Trusted hardware on the server
  - OTP code verified inside trusted environment
  - OTP key never leaves the hardware



Problems:
1. Is OTP code fresh?
2. Is OTP generated for correct domain (not phishing)?

Possible password replacements

- Cambridge's TR – wide range of possibilities listed
  - *The quest to replace passwords: a framework for comparative evaluation of Web authentication schemes*
  - https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-817.pdf
- Many different possibilities, but passwords are cheap to start with, a lot of legacy code exists and no mechanism offers all benefits

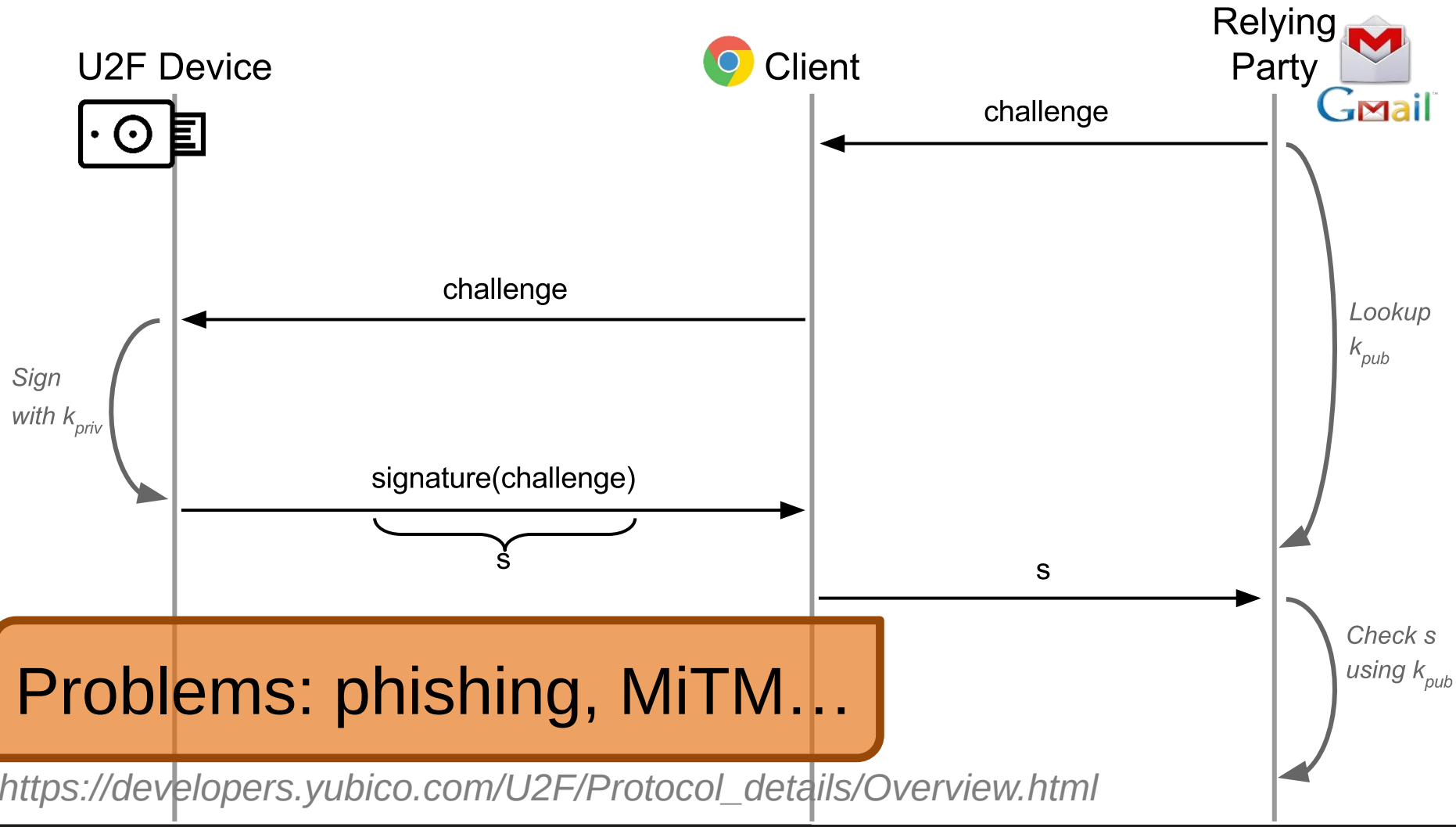
Mandatory reading: UCAM-CL-817

- At least chapters: II. Benefits, V. Discussion
- Whole report is highly recommended

IDEA: REPLACE PASSWORD BY SMARTCARD WITH ASYMMETRIC KEYPAIR, CHALLENGE-RESPONSE PROTOCOL AND PREVENT PHISHING

FIDO U2F PROTOCOL

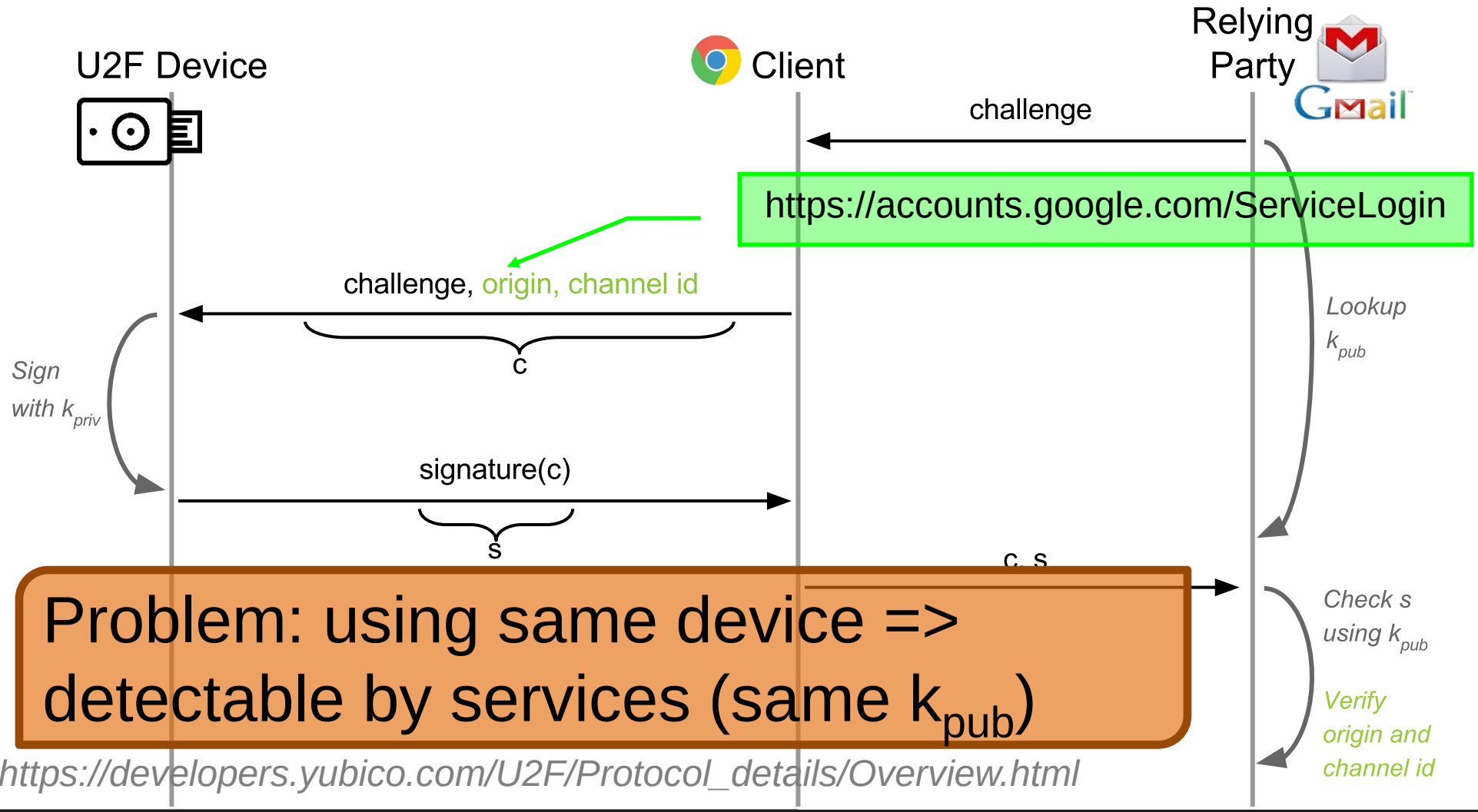
Revision 1: ECC-based challenge-response



Problems: phishing, MiTM…

https://developers.yubico.com/U2F/Protocol_details/Overview.html

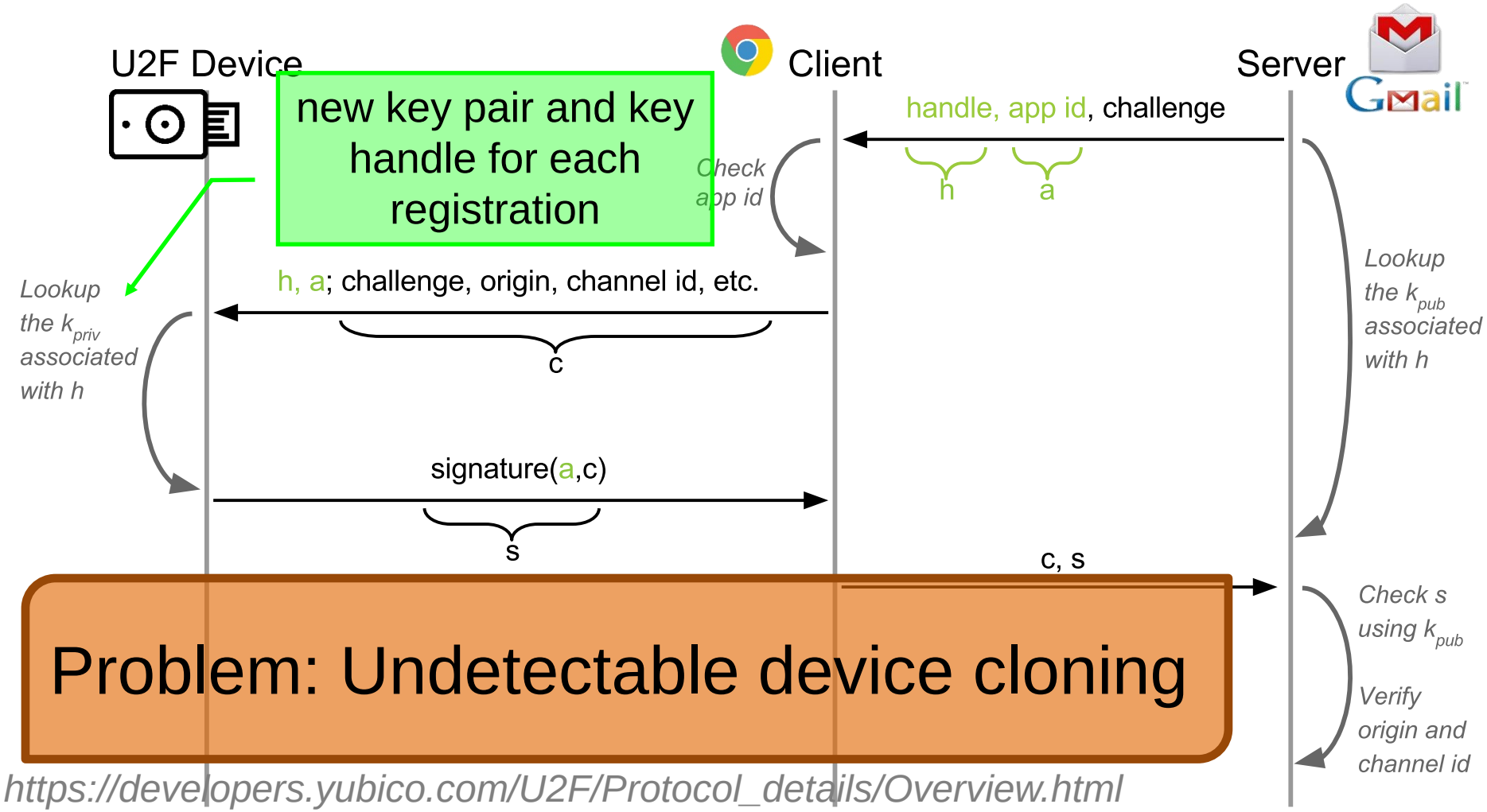
Revision 2: URI + TLS channel id added



Problem: using same device => detectable by services (same k_{pub})

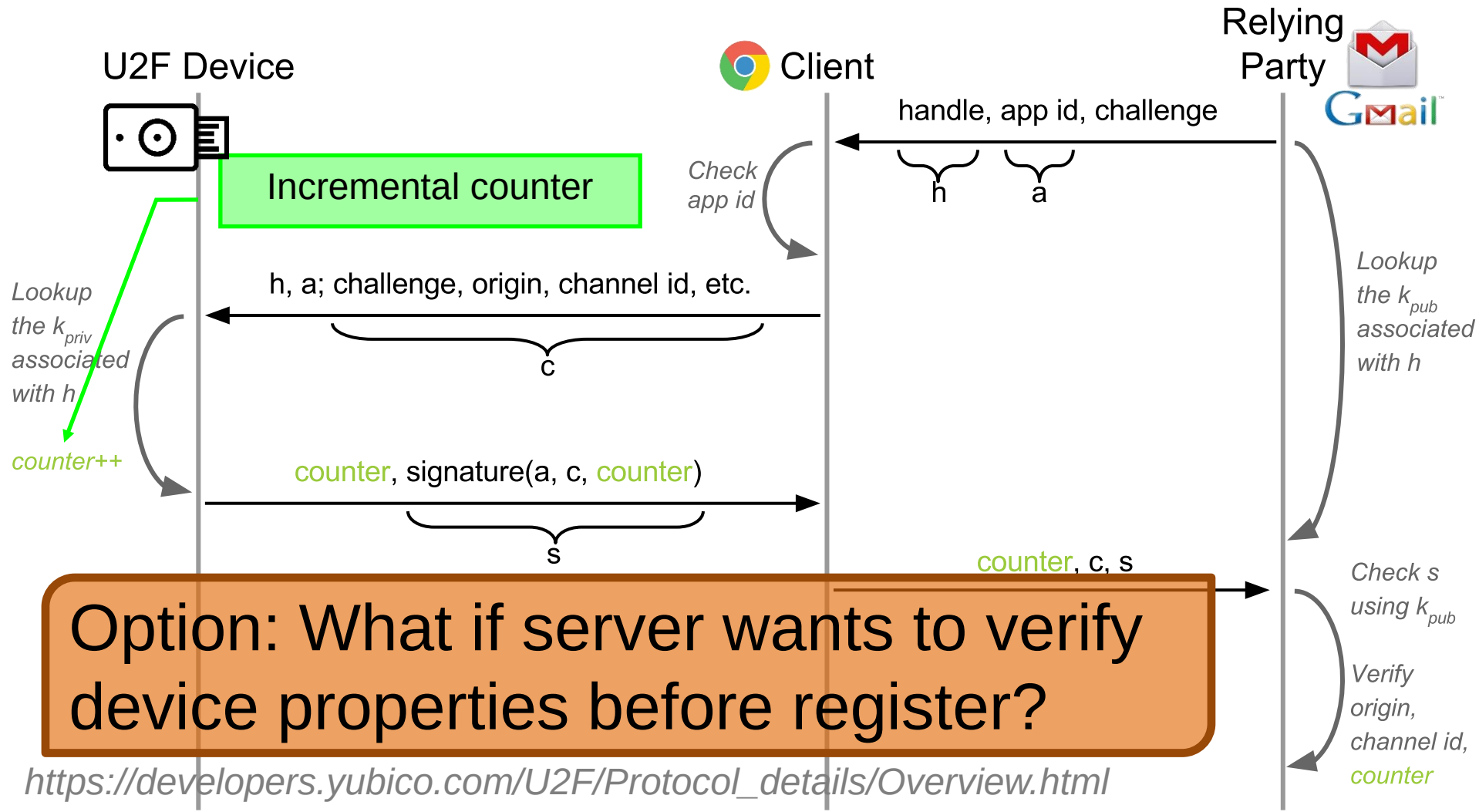
https://developers.yubico.com/U2F/Protocol_details/Overview.html

Revision 3: Application-specific key added

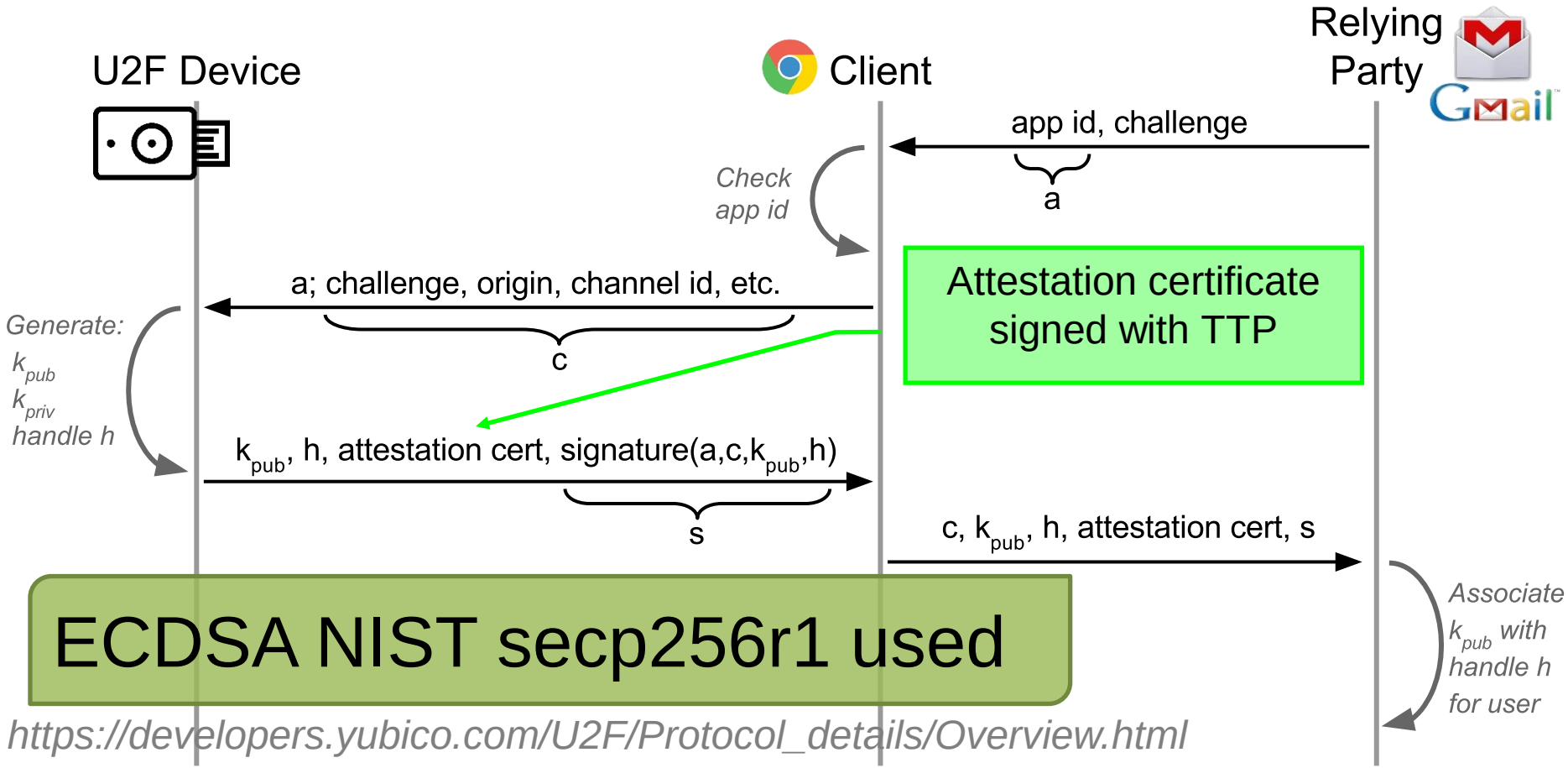


https://developers.yubico.com/U2F/Protocol_details/Overview.html

Revision 4: Authentication counter added



Revision 5: Device attestation added



ECDSA NIST secp256r1 used

https://developers.yubico.com/U2F/Protocol_details/Overview.html

FIDO U2F – current state

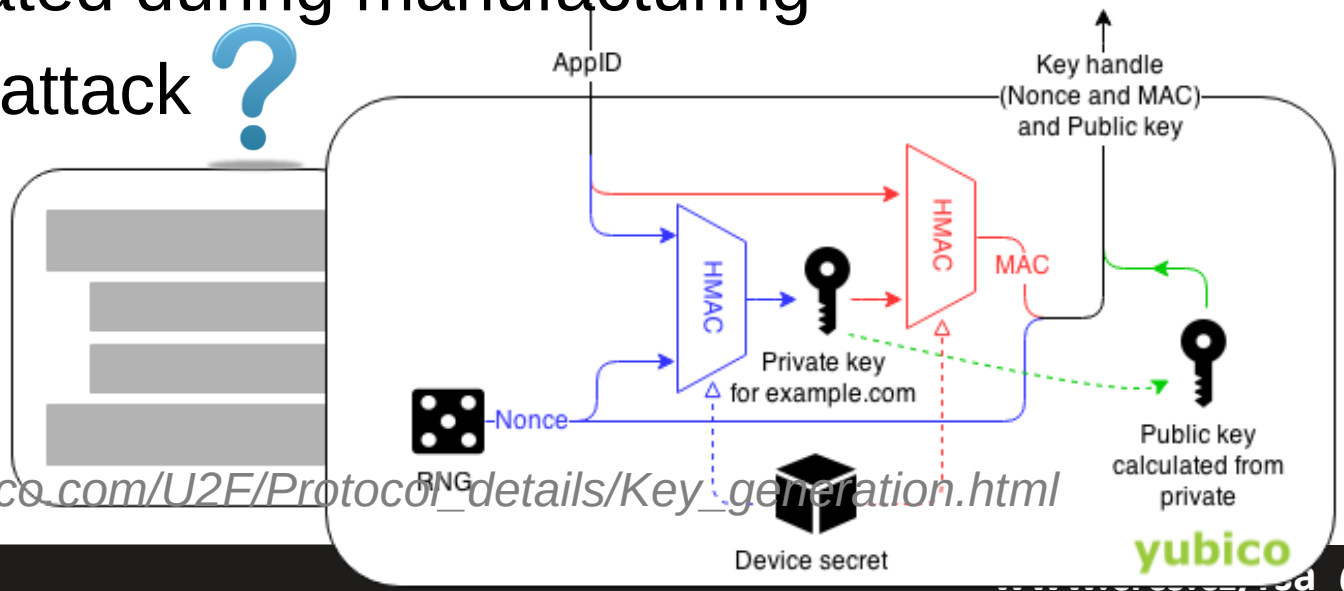
- FIDO alliance of major companies
- U2F → FIDO2 project (more than “just” U2F)
- Original U2F protocol extended and moved under W3 as WebAuthn
  - https://www.w3.org/TR/webauthn/
- Large selection of tokens now available (including open-hardware)
- Android added systematic support for FIDO U2F (02/2019)
  - Android phone acts as U2F token
  - https://www.wired.com/story/android-passwordless-login-fido2
- Google Smart Lock app on iOS uses secure enclave and acts as FIDO token
- Since iOS 13.3. USB, NFC, and Lightning FIDO2-compliant security keys in Safari browser

FIDO U2F devices

- Why have button? Is missing display problem?
- Recent problem: direct WebUSB API in Chrome
  - Malware bypass U2F API checking the URL
  - Legitimate URL is send from malicious page
  - https://www.wired.com/story/chrome-yubikey-phishing-webusb/
  - APDU-level communication: https://npmccallum.gitlab.io/post/u2f-protocol-overview/
- Well known is Yubikey, but open-source hardware and/or software-only implementations also possible
  - https://github.com/conorpp/u2f-zero
  - https://github.com/solokeys/solo

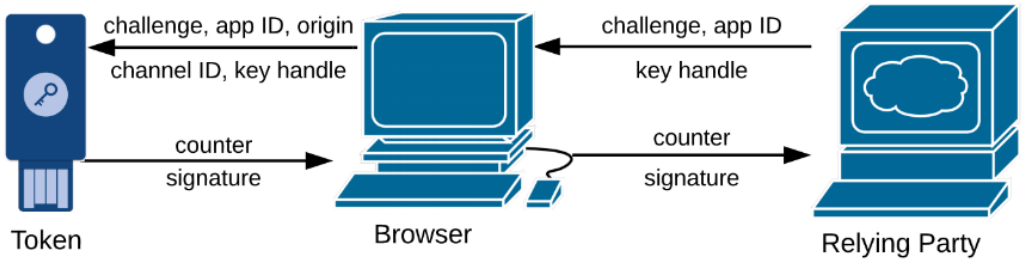
Always dig for implementation details

- How are ECC keys generated and stored?
- Yubikey saves storage memory by deriving ECC private keys from master secret instead of randomly generating new one
  - Possible as the ECC private key is random value
- Device secret generated during manufacturing
- What is the possible attack ?



https://developers.yubico.com/U2F/Protocol_details/Key_generation.html

True2F FIDO U2F token



- Yubikey 4 has single master key
  - To efficiently derive keypairs for separate Relying parties (Google, GitHub…)
  - Inserted during manufacturing phase (what if compromised?)
- Additional SMPC protocols (protection against backdoored token)
  - Secure Multi-Party Computation (SMPC) will be covered later
  - Verifiable insertion of browser randomness into final keypairs
  - Prevention of private key leakage via ECDSA padding

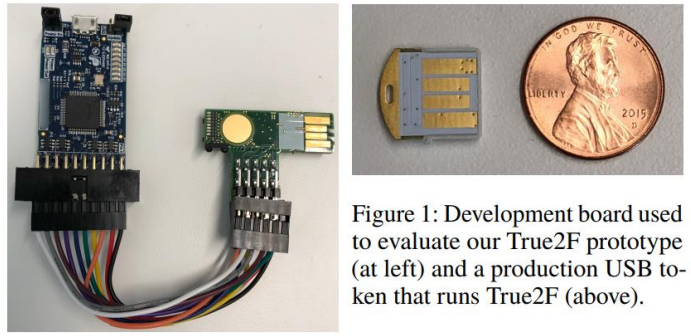


Figure 1: Development board used to evaluate our True2F prototype (at left) and a production USB token that runs True2F (above).

- Backward-compatible (Relying party, HW)
- Efficient: 57ms vs. 23ms to authenticate

WebAuthn

- An API for accessing Public Key Credentials Level 1
- https://www.w3.org/TR/webauthn/
- Similar, but more complex standard than U2F => expect additional problems (not yet scrutinized enough)

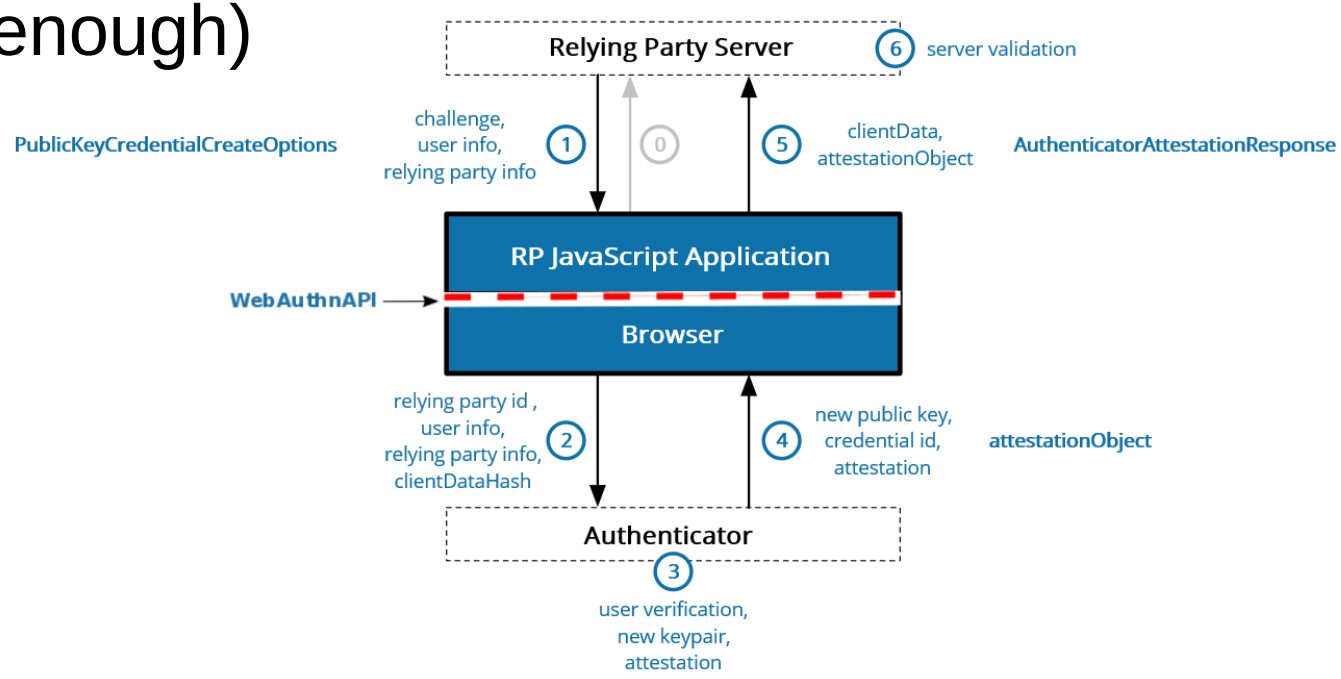


***Figure 1** Registration Flow*

Summary

- Passwords have multiple issues, but are hard to be replaced
- Major server-side breaches now very common
- Important to use passwords securely (guidelines)
- One-time passwords and tokens getting more used
- Password manager with synchronization over multiple devices is not straightforward, but doable (e.g., Apple's iCloud Keychain)
- Mandatory reading: UCAM-CL-817
  - At least chapters: II. Benefits, V. Discussion
  - Whole report is highly recommended

Top questions (1)

P PetrS 0

Is my password brute-force-able if consists of 9 printable characters?

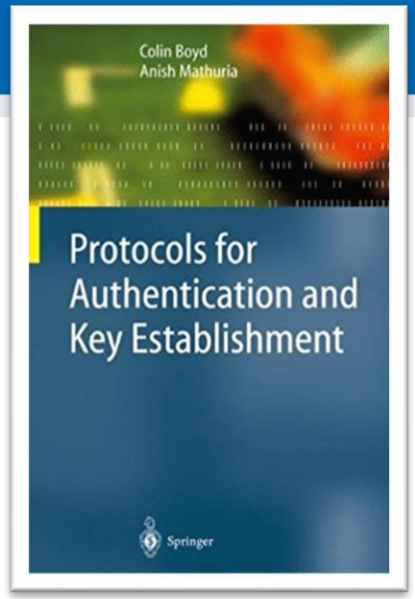
Join at

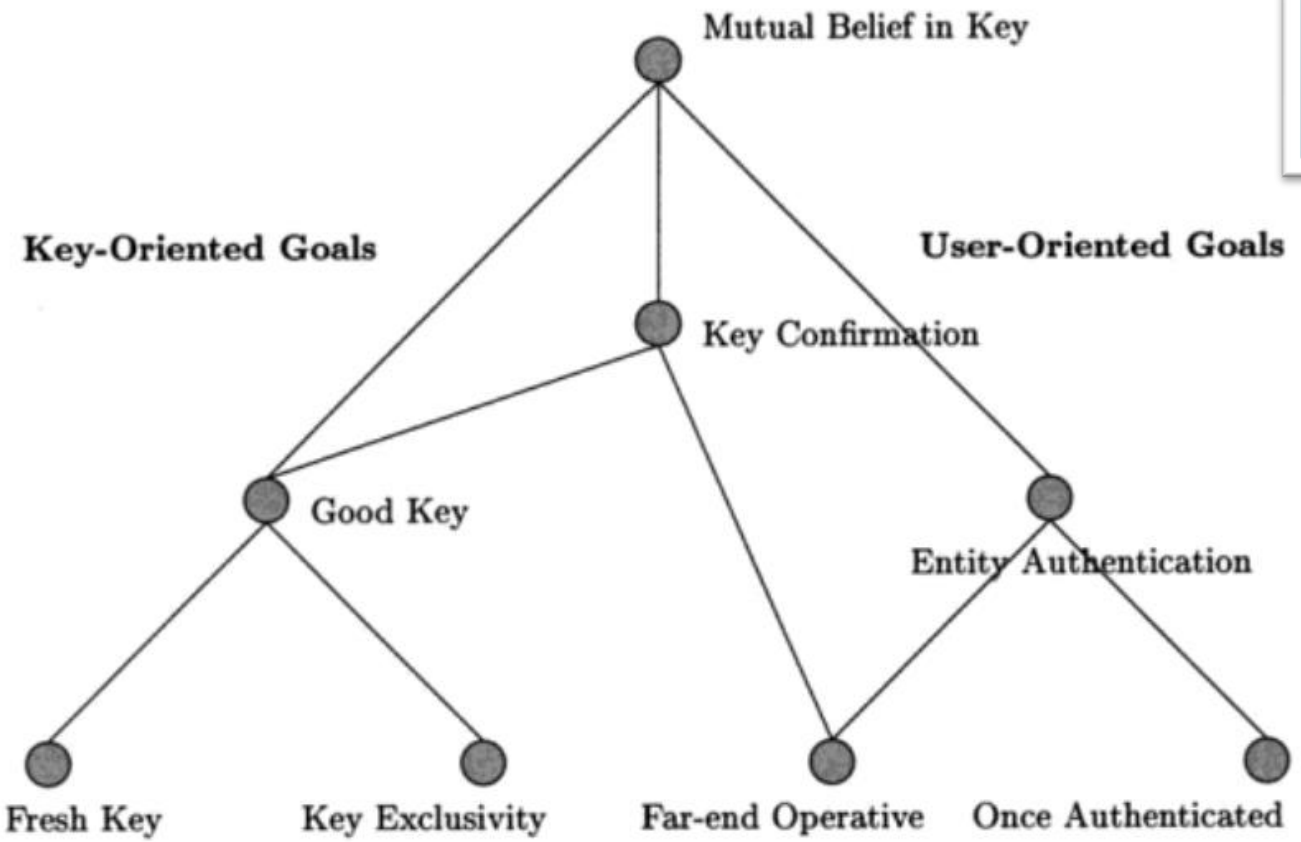
slido.com

#pv204_2021

- **Place/upvote questions in slido while listening to lecture video**
- **We will together discuss these during every week lecture Q&A (every Monday, 17-18:00)**

Hierarchy of authentication and key establishment goals





Protocols for Authentication and Key Establishment *By Colin Boyd, Anish Mathuria*

Common (mis-)Assumptions

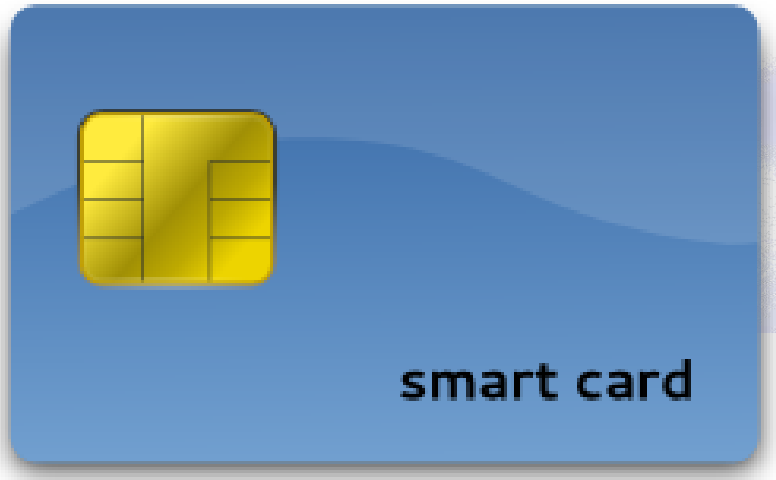
1. User has strong password
2. Server/service is hard to compromise
3. User have unique passwords
4. Different authentication channels are independent
5. Recovery

Human is still the weakest link

Google:
Sfdlk2c&432mo%
Skype:
*(&21mefd872!&

More than 60% of users have weak passwords

Hardware tokens



SmartArchitects's Password sCrib

Hack-a-Day's Mooltipass

Evergreen password lock

Price, usability, compatibility…

User has strong password

nakedsecurit

Award-winning news, opinion, advice and research from SOPHOS

malware mac facebook android vulnerability data loss priva

◀ Millions of LinkedIn passwords repo... Microsoft speaks out on Fla

LinkedIn confirms hack, over 60% o stolen passwords already cracked

by Chester Wisniewski on June 6, 2012 | 21 Comments

FILED UNDER: Data loss, Featured, Privacy, Social networks

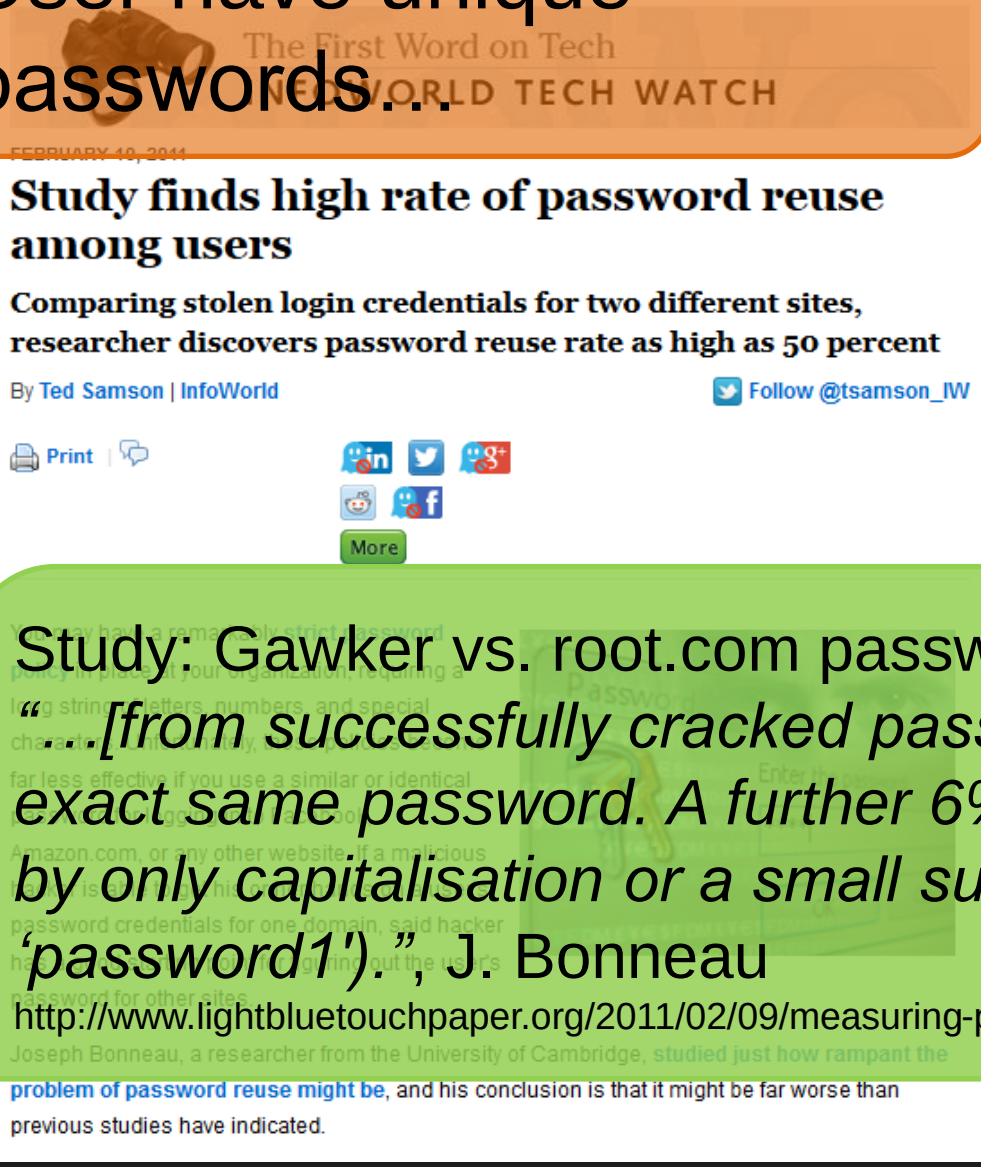
LinkedIn has confirmed that some of the password hashes that we online do match users of its service. They have also stated that pa that are reset will now be stored in salted hashed format.

LinkedIn Password Leak: The Top 30 Passwords Cracked (INFOGRAPHIC)

The Huffington Post | By Sara Gates

Posted: 06/08/2012 3:52 pm

User have unique passwords...



The First Word on Tech
INFOWORLD TECH WATCH

FEBRUARY 10, 2011

Study finds high rate of password reuse among users

Comparing stolen login credentials for two different sites, researcher discovers password reuse rate as high as 50 percent

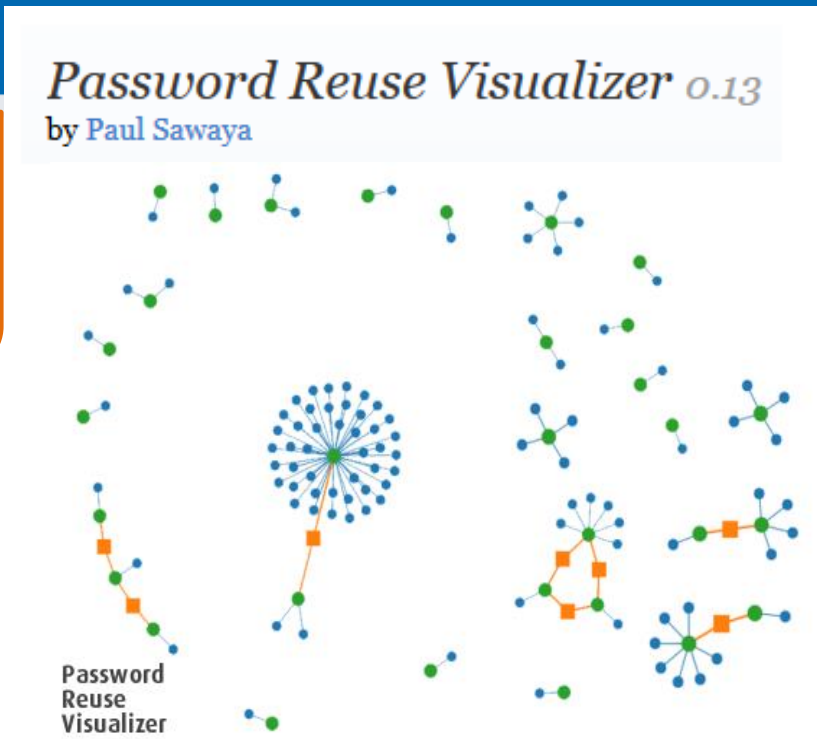
By Ted Samson | InfoWorld

Follow @tsamson_IW

Print

More

Joseph Bonneau, a researcher from the University of Cambridge, studied just how rampant the problem of password reuse might be, and his conclusion is that it might be far worse than previous studies have indicated.



Study: Gawker vs. root.com passwords leak

"...[from successfully cracked passwords] 76% used the exact same password. A further 6% used passwords differing by only capitalisation or a small suffix (e.g. 'password' and 'password1').", J. Bonneau

http://www.lightbluetouchpaper.org/2011/02/09/measuring-password-re-use-empirically/

Service is hard to compromise?

nakedsecurity

Award-winning news, opinion, advice and research from SOPHOS

malware mac facebook android vulnerabil

Search Results for:

eBay becomes the latest own up to a password bre

by Paul Ducklin on May 21, 2014 | Leave a comment

eBay has admitted to a dat your password, now!

Scribd, "world's largest o admits to network intrusi breach

by Paul Ducklin on April 5, 2013 | 2 Comments

San Francisco-based docu admitted to a network intru

Details are scant, but fortu by the company suggests t are at risk...

Australian Defence Force Academy in stinkingly bad password breach

by Paul Ducklin on December 11, 2012 | 6 Comments

The Australian Defence Force Academy is the latest high-profile organisation to become embroiled in a data breach.

The breach revealed names, birthdates, and some...well, some stinkingly bad passwords. Find out more

DreamHost warns customers of p password breach

by Graham Cluley on January 23, 2012 | 7 Comments

A database server at DreamHost is illegal a hacker, and the passwords of some cus have been compromised.

Trapster hack: millions warned of possible password breach

by Graham Cluley on January 20, 2011 | Leave a comment

Trapster is notifying its 10 million users tha usernames and passwords may have fall hands of hackers.

AOL Mail accounts breached, users advised to change passwords

by John Zorabedian on April 29, 2014 | 7 Comments

AOL users, change your passwords. AOL said it is investigating a large-scale breach of AOL Mail accounts in which user passwords, security questions, mail addresses, and contact lists were compromised.

Kickstarter breached - change your passwords

by Mark Stockley on February 16, 2014 | 2 Comments

Hackers gained unauthorised access to crowdfunding site kickstarter.com earlier this week. Compromised details include usernames, email addresses, mailing addresses, phone numbers and password hashes. Kickstarter users should change their passwords immediately.

Services follow the best security principles

ars technica

MAIN MENU · MY STORIES: 25 · FORUMS · SUBSCRIBE · JOBS

RISK ASSESSMENT / SECURITY & HACKTIV

Hackers expose 453,000 credentials allegedly taken from Yahoo service (Updated)

SQL injection retrieves user names and passwords stored in plaintext.

by Dan Goodin - July 12 2012, 6:53am CEST

HACKING · THE

Hackers posted what appear to be login credentials for more than 453,000 user accounts tha
said they retrieved in plaintext from an unidentified service on Yahoo.

PC NEWS / OPINIONS / FEATURES / DEALS / HOW-TO / BUSINESS / SUBSC

ALL REVIEWS · LAPTOPS / TABLETS / PHONES / APPS / SOFTWARE / SECURITY

Home / Reviews / Software / Mobile Apps / Starbucks App Saves Usernames, Passwords in Plain Text

Starbucks App Saves Usernames, Passwords in Plain Text

BY STEPHANIE MLOT JANUARY 16, 2014 10:38AM EST COMMENTS

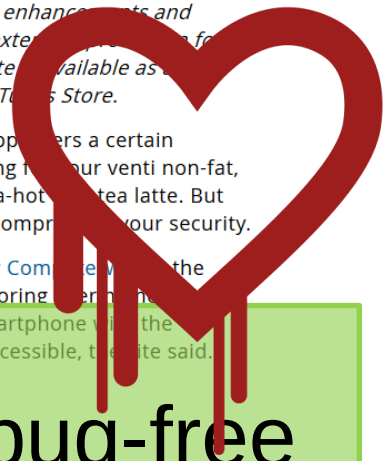
Starbucks has been storing user names, email addresses, and passwords in clear text—where anyone with a smartphone, USB cable, and computer can see them.

0 SHARES

UPDATE: *Starbucks on Friday released an updated version of its Starbucks Mobile App for iOS, with additional performance enhanc and safeguards, promising exte mobile users. The update ailable as download in the Apple iT Store.*

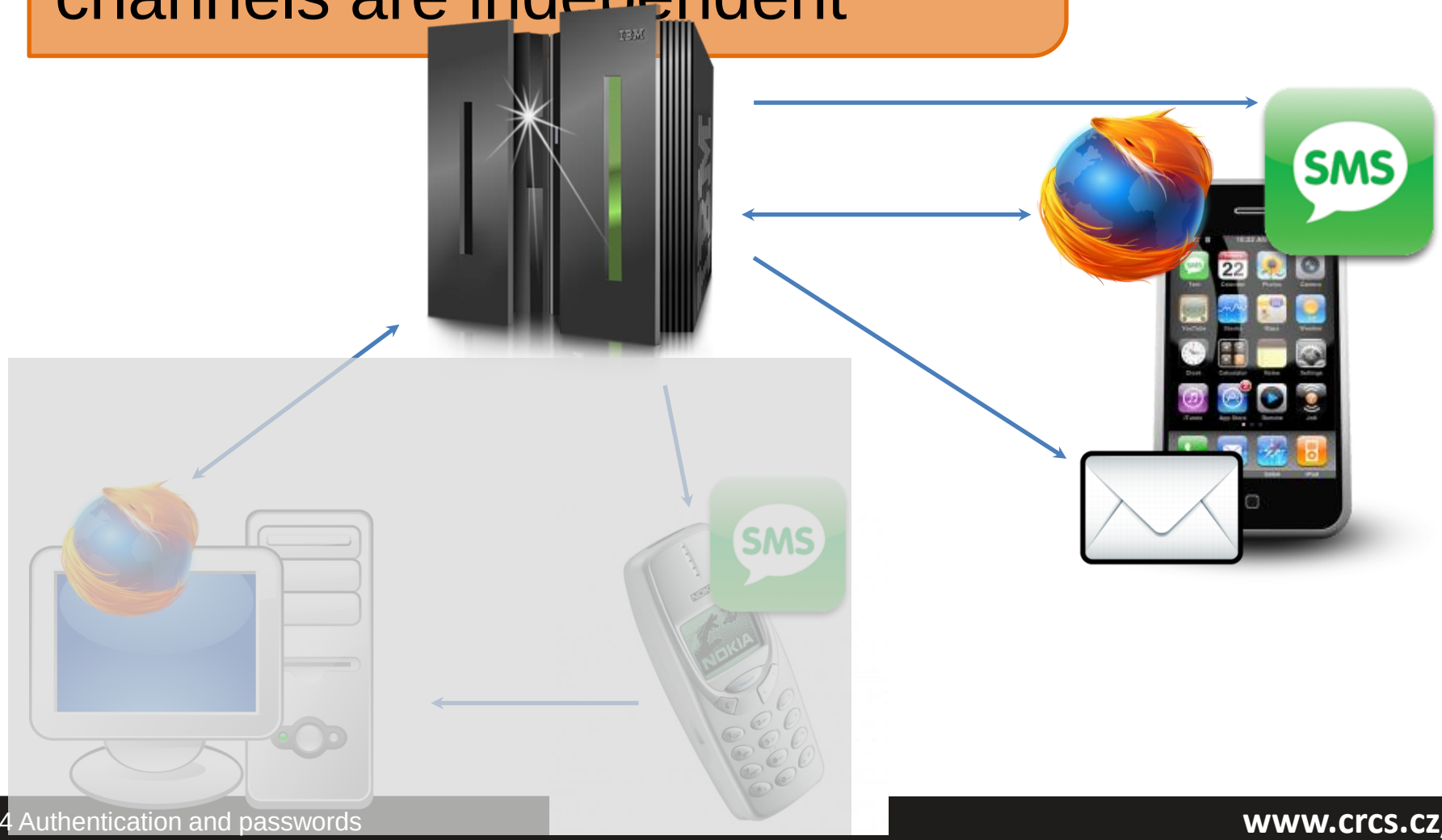
The Starbucks mobile app rs a certain convenience when paying f ur venti non-fat, no-foam, six-pump extra-hot tea latte. But turns out, it could also compr our security.

According to a report by Com the massively popular international coffee chain has been storing addresses, and passwords in clear text. Connecting a smartphone the Starbucks app installed to a PC, the password is easily accessible, t ite said.



Service implementation is correct and bug-free

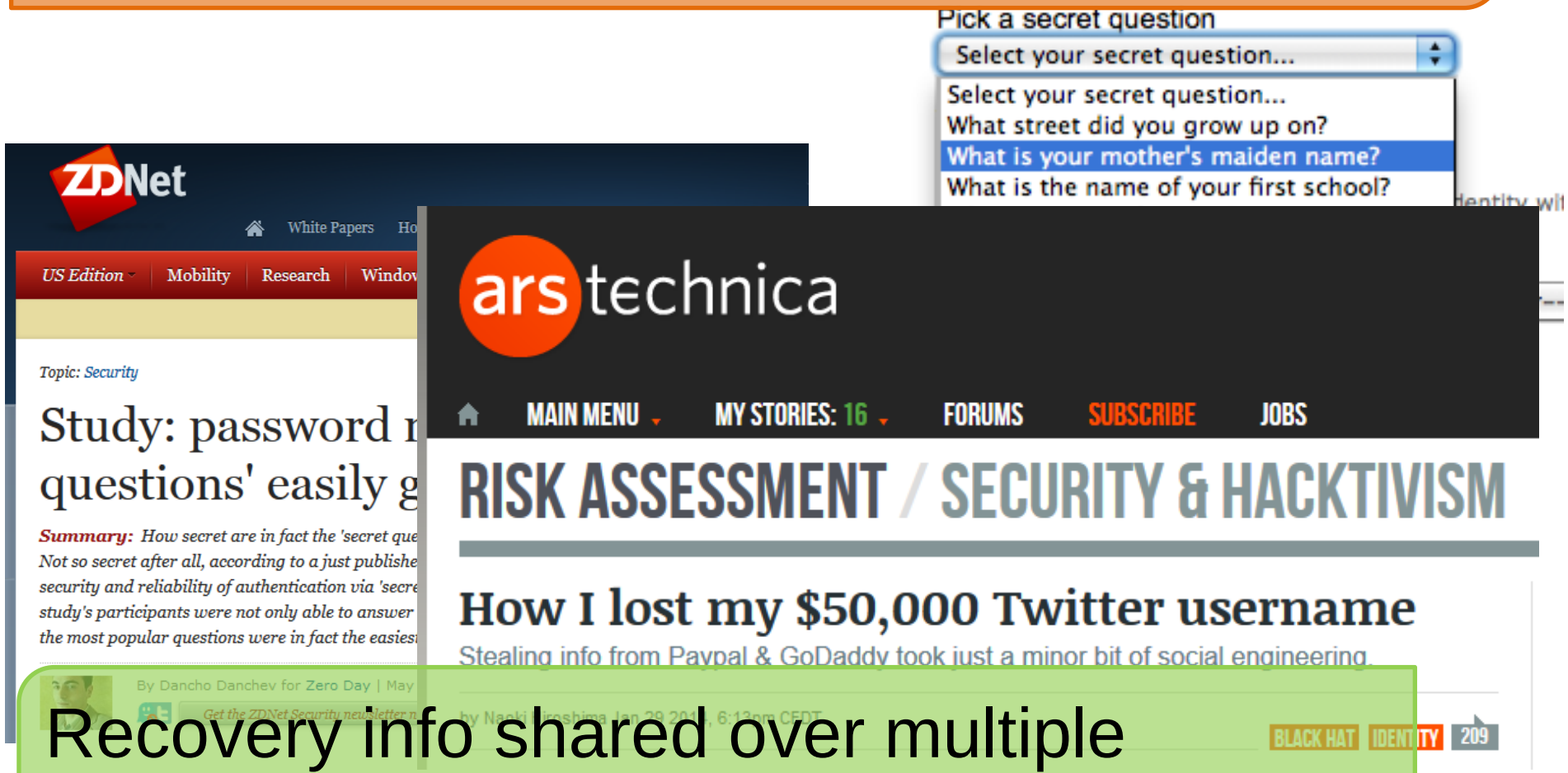
Different authentication channels are independent



Security can be maintained “forever”

Allow for some form of risk management

Access recovery is as secure as primary one



Recovery info shared over multiple services…

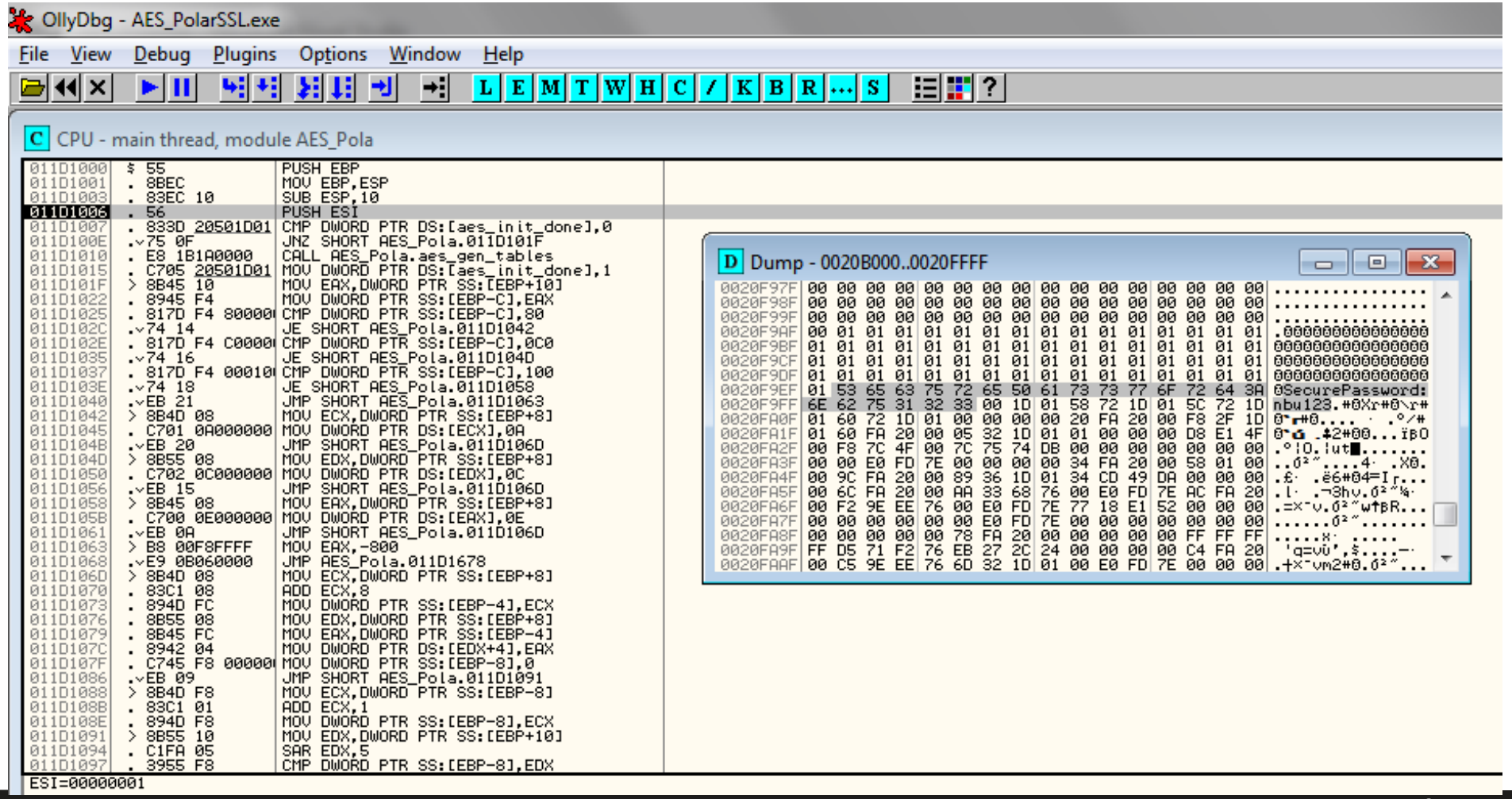
Password cracking defenses

- Don’t transmit or store in plaintext
- Process password on client, transmit only digest
- Don’t encrypt, hash instead
- Use salt to prevent rainbow tables attack
- Use memory-hard KDF algorithms
  - To slow down custom build hardware
  - Use strong KDF to derive keys (PBKDF2→Argon2)
- Use password-authenticated key exchange instead of password check

Handling passwords in source code

- Limiting memory exposure
  - Load only when needed
  - Erase right after use
  - Pass by reference / pointer to prevent copy in memory
  - Derive session keys
- Don’t hardcode password into application binary
- Nice presentation (K. Kohli, examples how NOT to): http://www.slideshare.net/amiable_indian/insecure-implementation-of-security-best-practices-of-hashing-captchas-and-caching-presentation

Hard-coded password might be visible both in application binary and memory



Alternative to hardcoded passwords/keys

- Don’t use passwords ☺
- Ask the user for a password
- Keep secrets in a separate file
- Encrypt stored secrets
- Store secrets in protected database
- Use already existing authentication credentials
- CERN guidelines
  - https://security.web.cern.ch/security/recommendations/en/password_alternatives.shtml

Group activity

- Form group of 3-4 members (mix, not your neighbours)
  - Introduce yourself with your name
- Discuss and write down on paper:
  - What method(s) you use for authentication (password…)
  - Is server using other authentication factor?
  - How you store the authentication secret? (brain-only…)
- Time limit: 5 minutes

- Now return back to your original seat (if you wish ☺)

Activity:

- Think about one or two surprising things from this lecture
- I want to hear at least 5 of these, tell me please ☺