PV204 Security technologies

Authentication and passwords

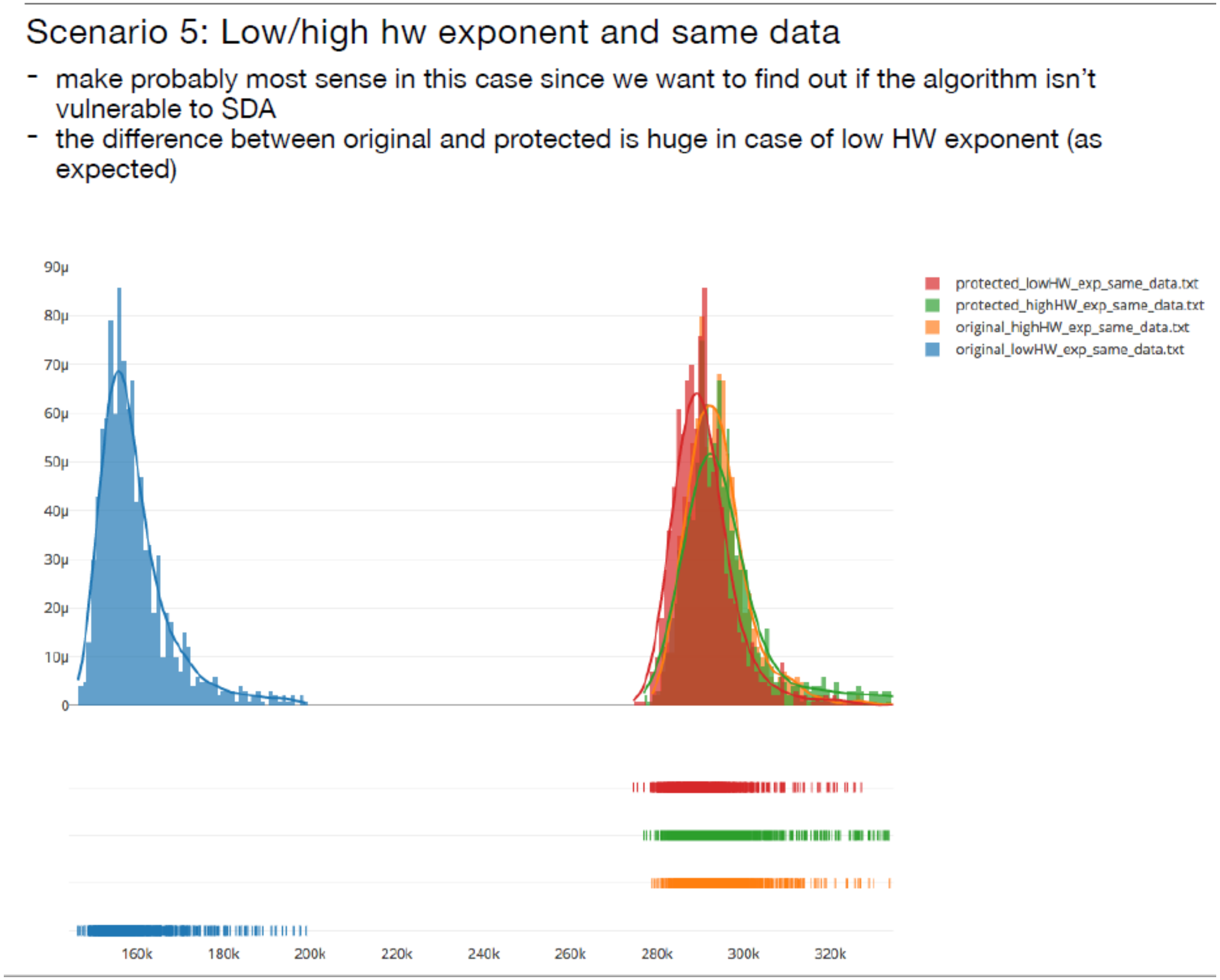
Petr Švenda svenda@fi.muni.cz
Faculty of Informatics, Masaryk University

CROCS
Centre for Research on
Cryptography and Security

How to approach homework I.

- What you should specify your configuration (replication)
  - live vs. dedicated machine → impact on measurement
  - list platform configuration, compilation flags used (for replication)
- Analysis
  - Good to print only shape of histogram instead of bars (visibility)
  - Good to print multiple histograms in single graph - better visibility
  - Don't compare only original to the protected version. More sense makes to compare different data/exponent for given version (e.g., protected)
- Good to test also with medium hamming weight
  - More spread in histogram with same data → harder to use Template attacks
  - Be aware what is included in timing - e.g., generation of masking r? Network jitter (can attacker model and subtract)?

Example solution (J. Masarik)



Scenario 5: Low/high hw exponent and same data

- make probably most sense in this case since we want to find out if the algorithm isn't vulnerable to SDA
- the difference between original and protected is huge in case of low HW exponent (as expected)

Conclusions for Homework I.

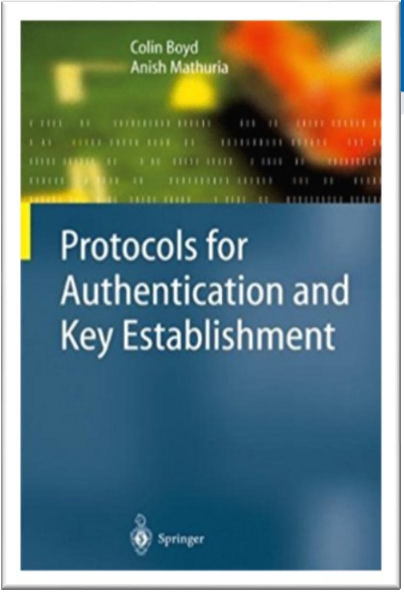
- Variability for the same data and key => noise in measurement (=> potentially harder to attack)
- Difference between any two measured values => possibility to use template attack
- Difference between data with low/mid/high hamming weight => some information is leaking
- Dependency of time on HW of private exponent may be possible to detect even for improved or blinded RSA
- Compiler can remove inserted protection (releases)

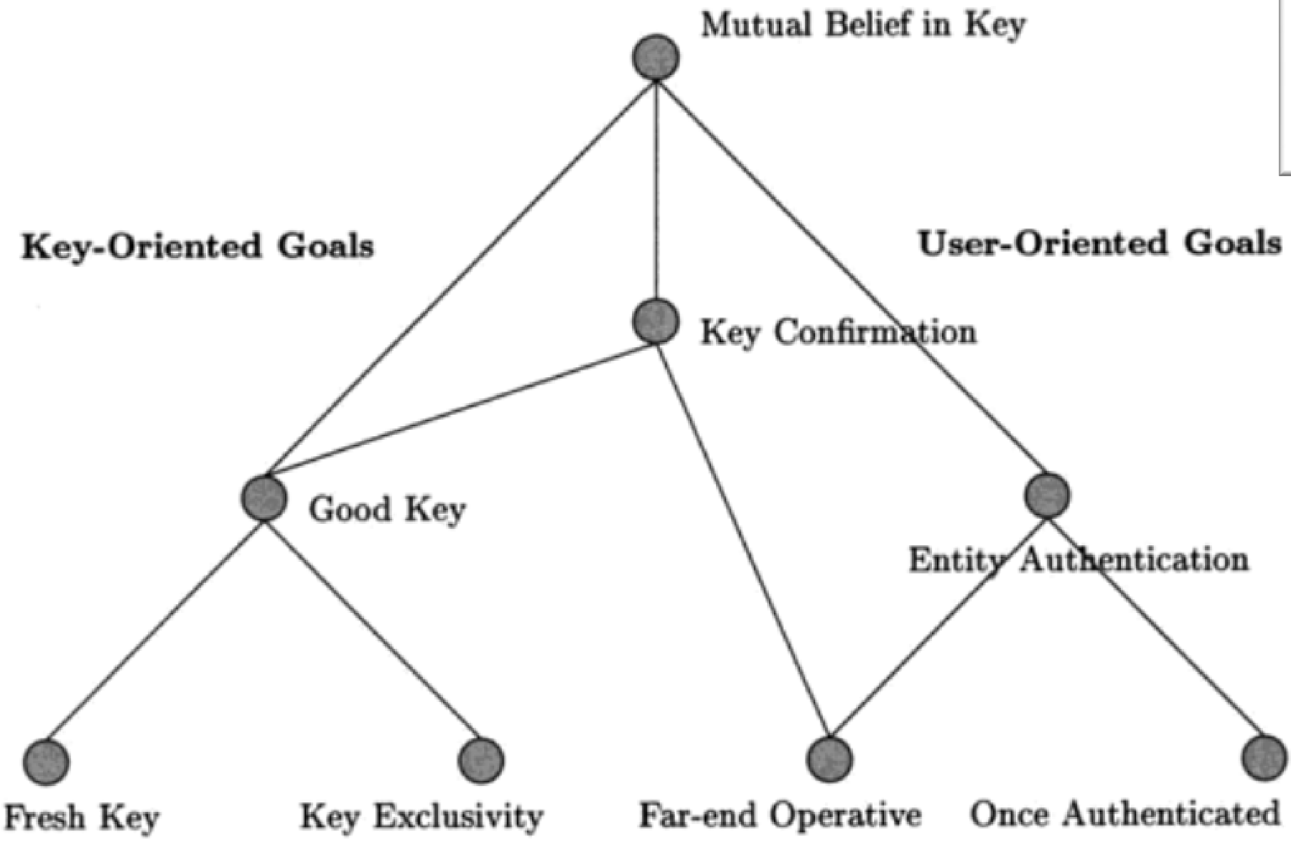
AUTHENTICATION & AUTHORIZATION

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 entity allowed to use resource to which is authorized

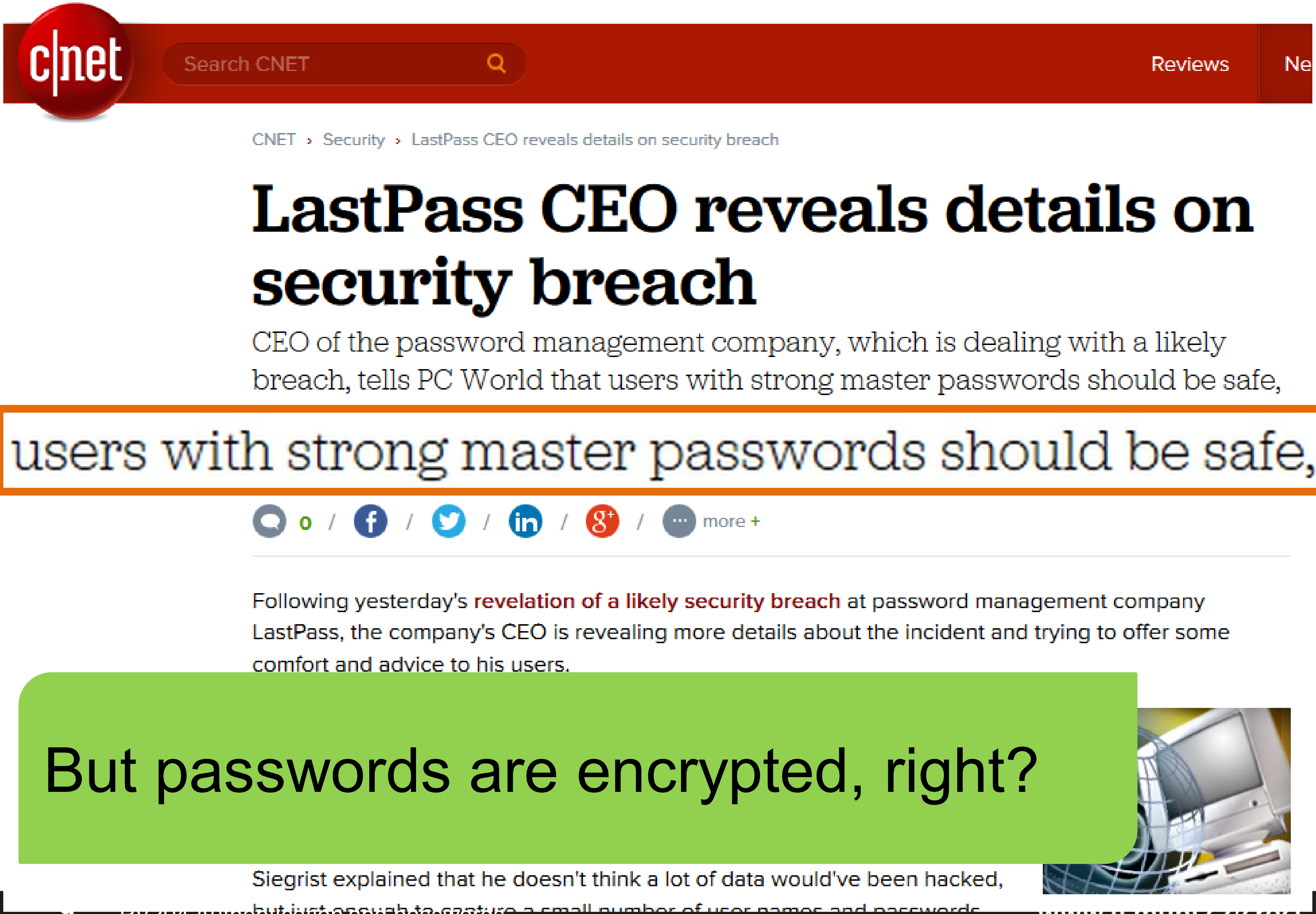
Hierarchy of authentication and key establishment goals





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

PASSWORDS



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.

But passwords are encrypted, right?

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.

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 target operation (e.g., authorization of request)
- …

Mode of usage for passwords

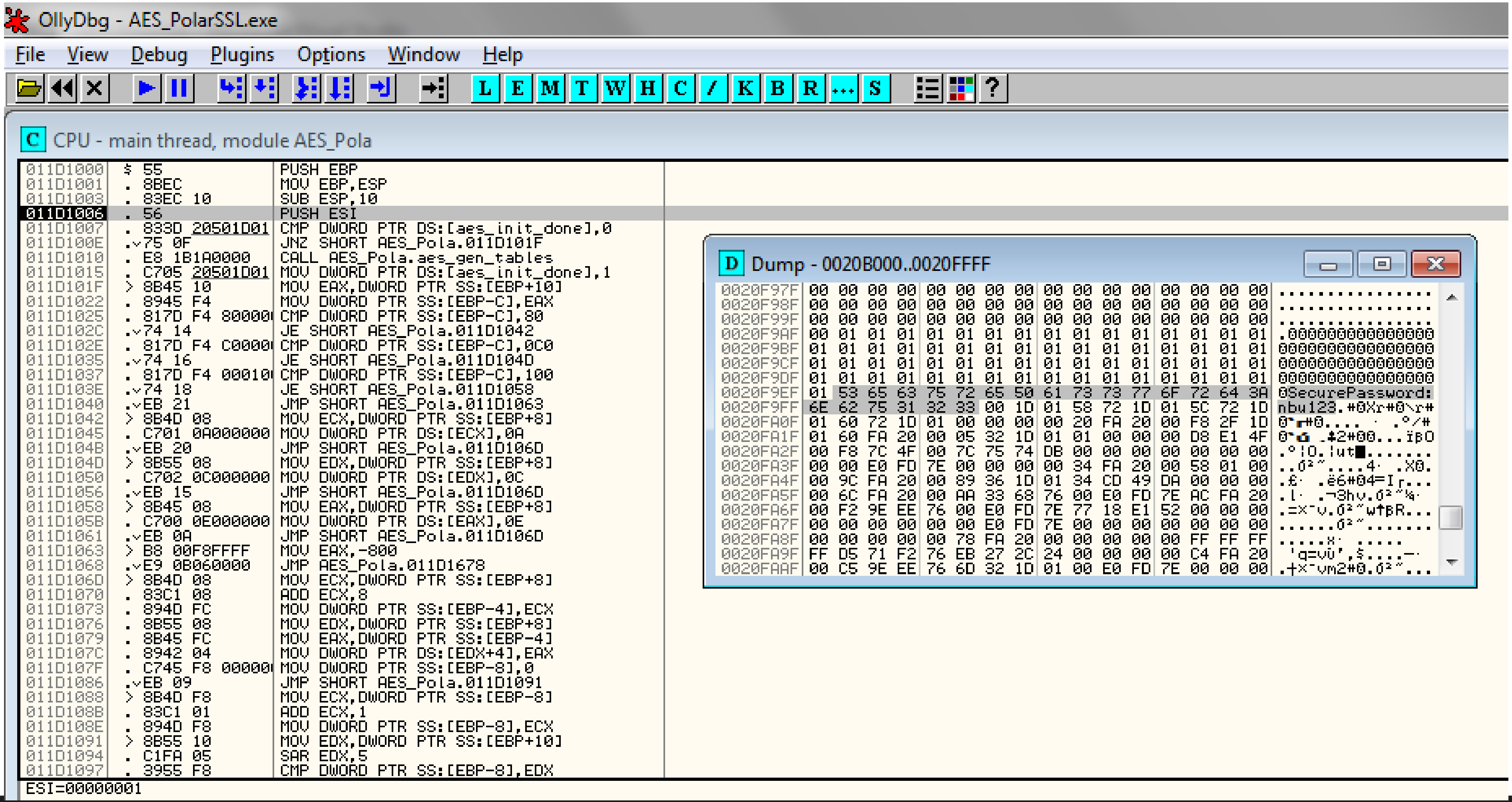
1. Verify by direct match
   - provided_password == expected_password?
   - Example: HTTP basic access authentication
   - Be aware of potential side-channels
2. Verify by derived value (hash(password | salt))
   - Be aware of rainbow tables and brute-force crackers
3. Derive key: Password → cryptographic key
   - Example: key = PBKDF2(password)
4. Use to establish authenticated key
   - Example: Password + Diffie-Hellman → authenticated key…

Where passwords can be compromised?

1. Database storage
   - Cleartext storage
   - Backup data (tapes)
   - Server compromise
2. Host machine (memory, history, cache)
3. Network transmission (network sniffer, proxy logs)
4. Hardcoded secrets (inside app binary)

- Difficult to detect compromise and change after the exposure

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



OllyDbg - AES_PolarSSL.exe
File View Debug Plugins Options Window Help
L E M T W H C / K B R ... S ?
CPU - main thread, module AES_Pola
Dump - 0020B000..0020FFFF

```
011D1000  $ 55            PUSH EBP
011D1001  . 8BEC          MOV EBP,ESP
011D1003  . 83EC 10       SUB ESP,10
011D1006  . 56            PUSH ESI
011D1007  . 833D 20501D01 CMP DWORD PTR DS:[aes_init_done],0
011D100E  .v75 0F         JNZ SHORT AES_Pola.011D101F
011D1010  . E8 1B1A0000   CALL AES_Pola.aes_gen_tables
011D1015  . C705 20501D01 MOV DWORD PTR DS:[aes_init_done],1
011D101F  > 8B45 10       MOV EAX,DWORD PTR SS:[EBP+10]
011D1022  . 8945 F4       MOV DWORD PTR SS:[EBP-C],EAX
011D1025  . 817D F4 80000 CMP DWORD PTR SS:[EBP-C],80
011D102C  .v74 14         JE SHORT AES_Pola.011D1042
011D102E  . 817D F4 C0000 CMP DWORD PTR SS:[EBP-C],0C0
011D1035  .v74 16         JE SHORT AES_Pola.011D104D
011D1037  . 817D F4 00010 CMP DWORD PTR SS:[EBP-C],100
011D103E  .v74 18         JE SHORT AES_Pola.011D1058
011D1040  .vEB 21         JMP SHORT AES_Pola.011D1063
011D1042  > 8B4D 08       MOV ECX,DWORD PTR SS:[EBP+8]
011D1045  . C701 0A000000 MOV DWORD PTR DS:[ECX],0A
011D104B  .vEB 20         JMP SHORT AES_Pola.011D106D
011D104D  > 8B55 08       MOV EDX,DWORD PTR SS:[EBP+8]
011D1050  . C702 0C000000 MOV DWORD PTR DS:[EDX],0C
011D1056  .vEB 15         JMP SHORT AES_Pola.011D106D
011D1058  > 8B45 08       MOV EAX,DWORD PTR SS:[EBP+8]
011D105B  . C700 0E000000 MOV DWORD PTR DS:[EAX],0E
011D1061  .vEB 0A         JMP SHORT AES_Pola.011D106D
011D1063  > B8 00F8FFFF   MOV EAX,-800
011D1068  .vE9 0B060000   JMP AES_Pola.011D1678
011D106D  > 8B4D 08       MOV ECX,DWORD PTR SS:[EBP+8]
011D1070  . 83C1 08       ADD ECX,8
011D1073  . 894D FC       MOV DWORD PTR SS:[EBP-4],ECX
011D1076  . 8B55 08       MOV EDX,DWORD PTR SS:[EBP+8]
011D1079  . 8B45 FC       MOV EAX,DWORD PTR SS:[EBP-4]
011D107C  . 8942 04       MOV DWORD PTR DS:[EDX+4],EAX
011D107F  . C745 F8 00000 MOV DWORD PTR SS:[EBP-8],0
011D1086  .vEB 09         JMP SHORT AES_Pola.011D1091
011D1088  > 8B4D F8       MOV ECX,DWORD PTR SS:[EBP-8]
011D108B  . 83C1 01       ADD ECX,1
011D108E  . 894D F8       MOV DWORD PTR SS:[EBP-8],ECX
011D1091  > 8B55 10       MOV EDX,DWORD PTR SS:[EBP+10]
011D1094  . C1FA 05       SAR EDX,5
011D1097  . 3955 F8       CMP DWORD PTR SS:[EBP-8],EDX
ESI=00000001
```

```
0020F97F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0020F98F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0020F99F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0020F9AF 00 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 .000000000000000
0020F9BF 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 0000000000000000
0020F9CF 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 0000000000000000
0020F9DF 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 0000000000000000
0020F9EF 01 53 65 63 75 72 65 50 61 73 73 77 6F 72 64 3A 0SecurePassword:
0020F9FF 6E 62 75 31 32 33 00 1D 01 58 72 1D 01 5C 72 1D nbu123.#0Xr#0\r#
0020FA0F 01 60 72 1D 01 00 00 00 00 20 FA 20 00 F8 2F 1D 0ˉr#0....  ·  .°/#
0020FA1F 01 60 FA 20 00 05 32 1D 01 01 00 00 00 D8 E1 4F 0ˉ· .‡2#00...ïβO
0020FA2F 00 F8 7C 4F 00 7C 75 74 DB 00 00 00 00 00 00 00 .°¦O.¦ut█.......
0020FA3F 00 00 E0 FD 7E 00 00 00 00 34 FA 20 00 58 01 00 ..ó²˜....4·  .X0.
0020FA4F 00 9C FA 20 00 89 36 1D 01 34 CD 49 DA 00 00 00 .£·  .ë6#04═I┌...
0020FA5F 00 6C FA 20 00 AA 33 68 76 00 E0 FD 7E AC FA 20 .l·  .¬3hv.ó²˜¼·
0020FA6F 00 F2 9E EE 76 00 E0 FD 7E 77 18 E1 52 00 00 00 .=×¯v.ó²˜w↑βR...
0020FA7F 00 00 00 00 00 00 E0 FD 7E 00 00 00 00 00 00 00 ......ó²˜.......
0020FA8F 00 00 00 00 00 78 FA 20 00 00 00 00 00 FF FF FF .....x·  .....
0020FA9F FF D5 71 F2 76 EB 27 2C 24 00 00 00 00 C4 FA 20  'q=vû',$....─·
0020FAAF 00 C5 9E EE 76 6D 32 1D 01 00 E0 FD 7E 00 00 00 .┼×¯vm2#0.ó²˜...
```

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, 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

ONE-TIME PASSWORDS

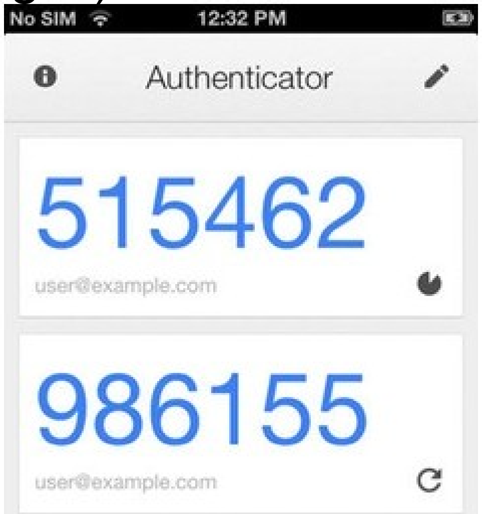
Recall: Problems associated with passwords

- 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

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



No SIM 12:32 PM

Authenticator

515462

user@example.com

986155

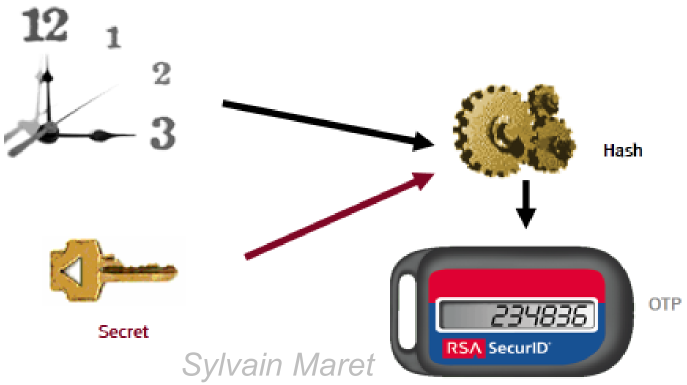
user@example.com

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



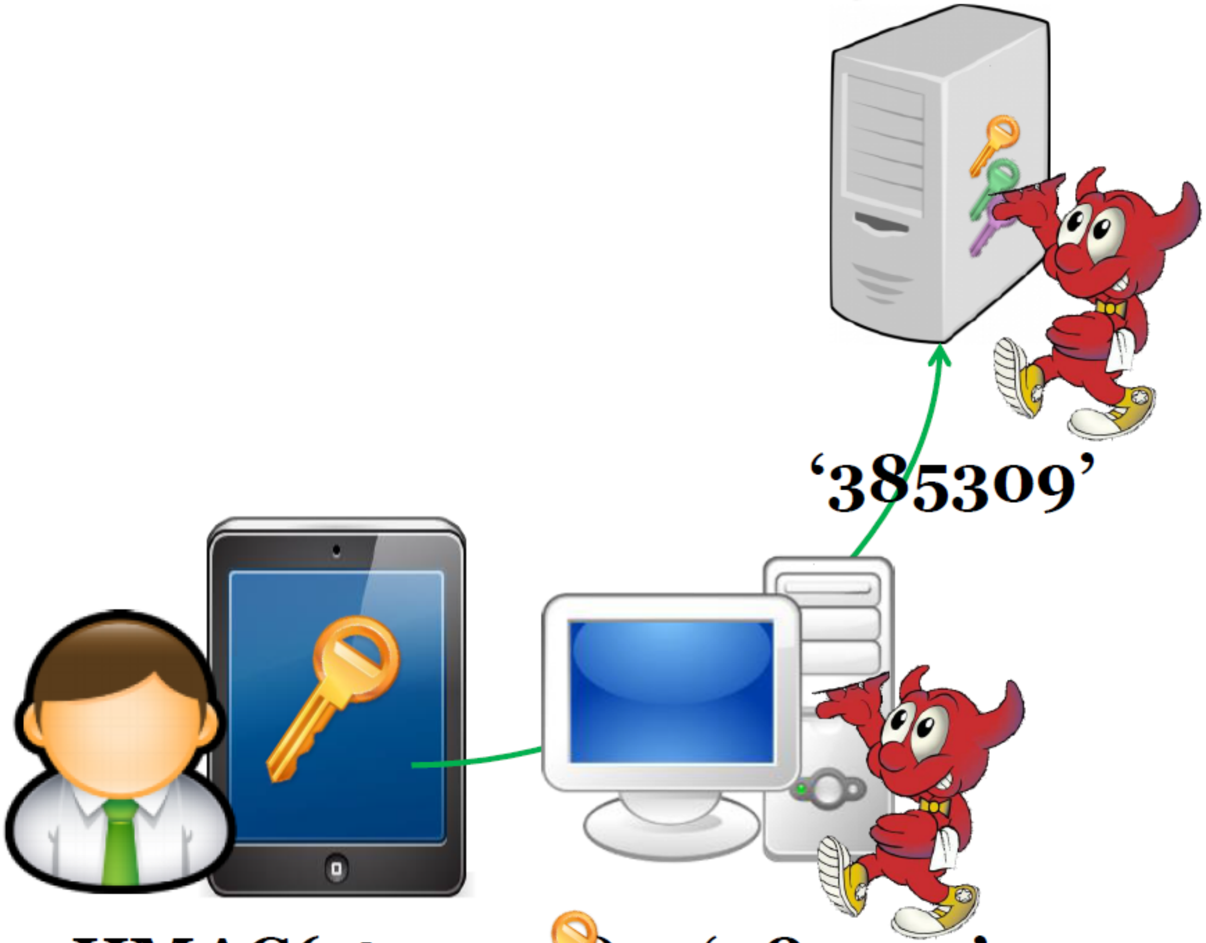
Sylvain Maret

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’?

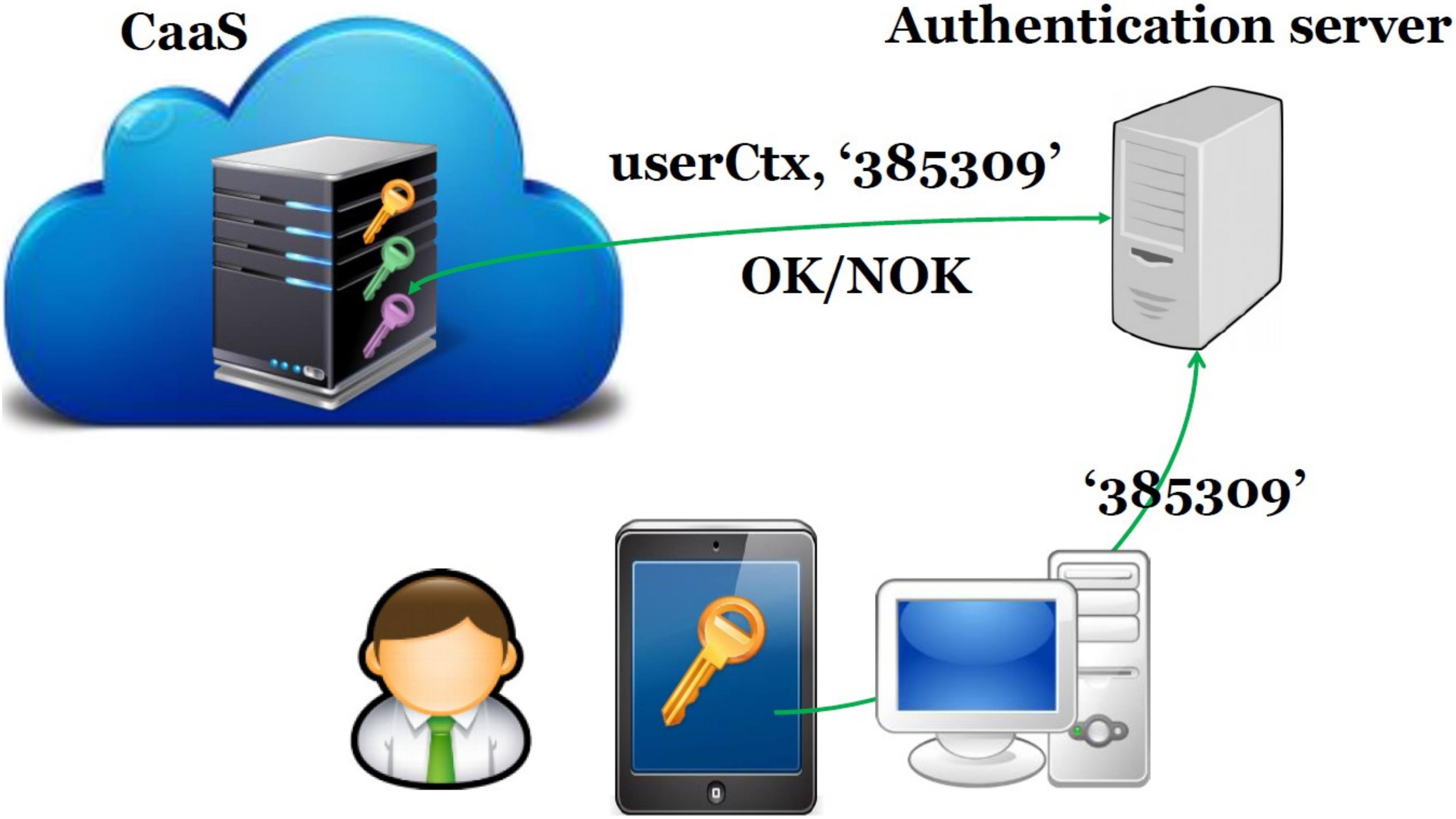


‘385309’

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



CaaS

Authentication server

userCtx, ‘385309’

OK/NOK

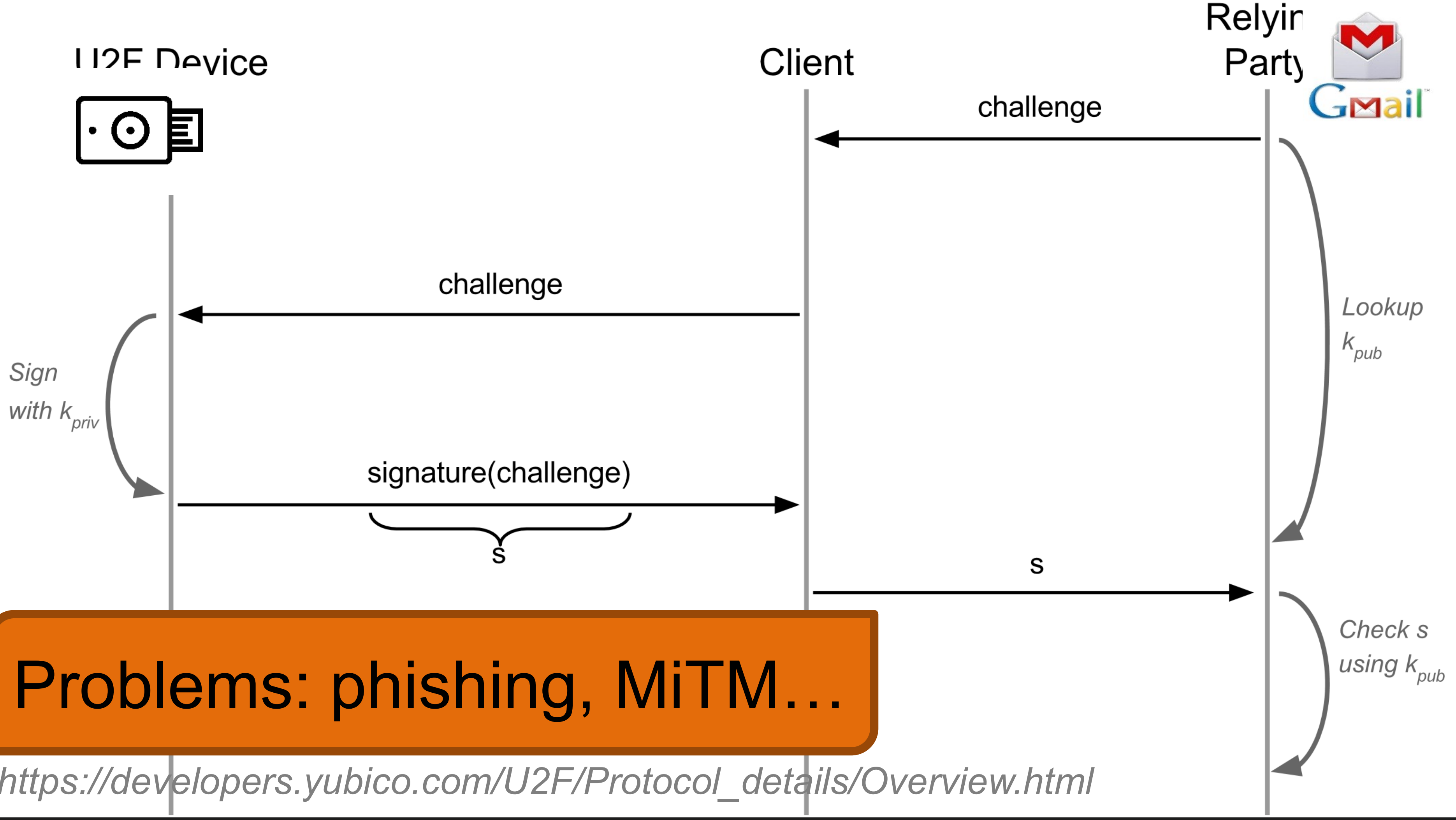
‘385309’

HOTP = HMAC(ctr++,🔑) = ‘385309’

PROBLEM: IS OTP CODE FRESH?

FIDO U2F PROTOCOL

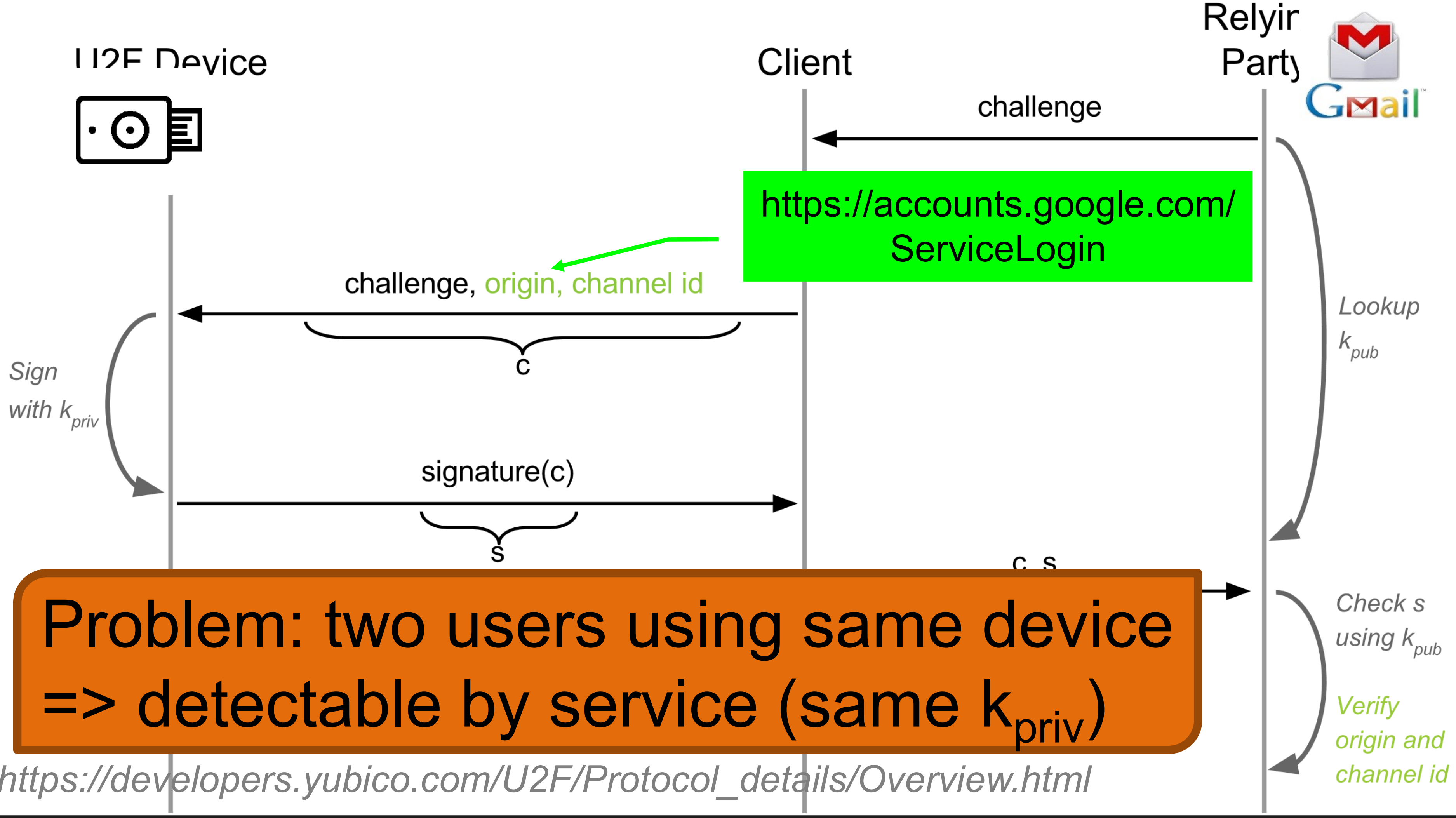
Revision 1: ECC-based challenge-response



Problems: phishing, MiTM…

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

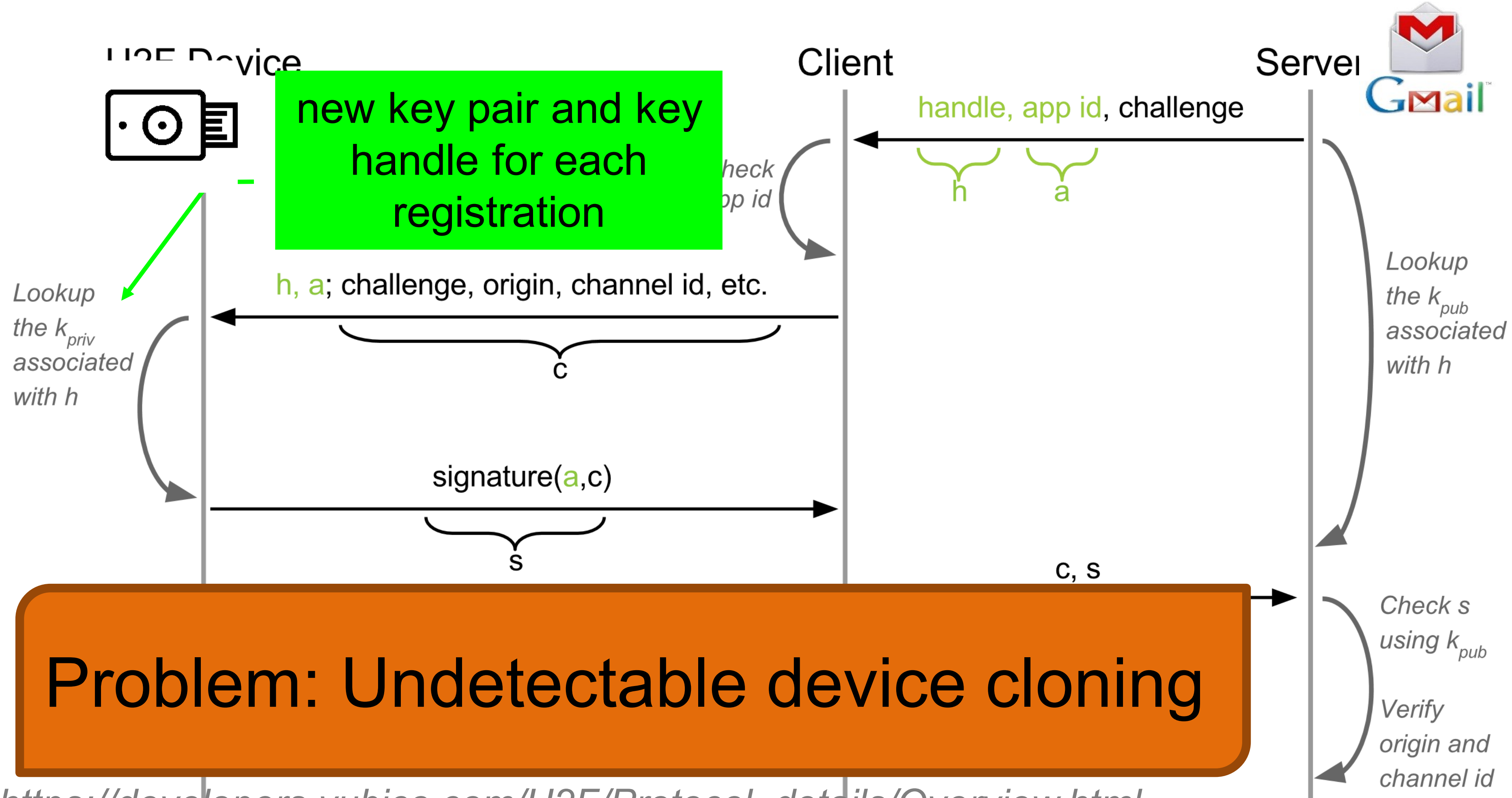
Revision 2: URI + TLS channel id added



Problem: two users using same device => detectable by service (same k_{priv})

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

Revision 3: Application-specific key added

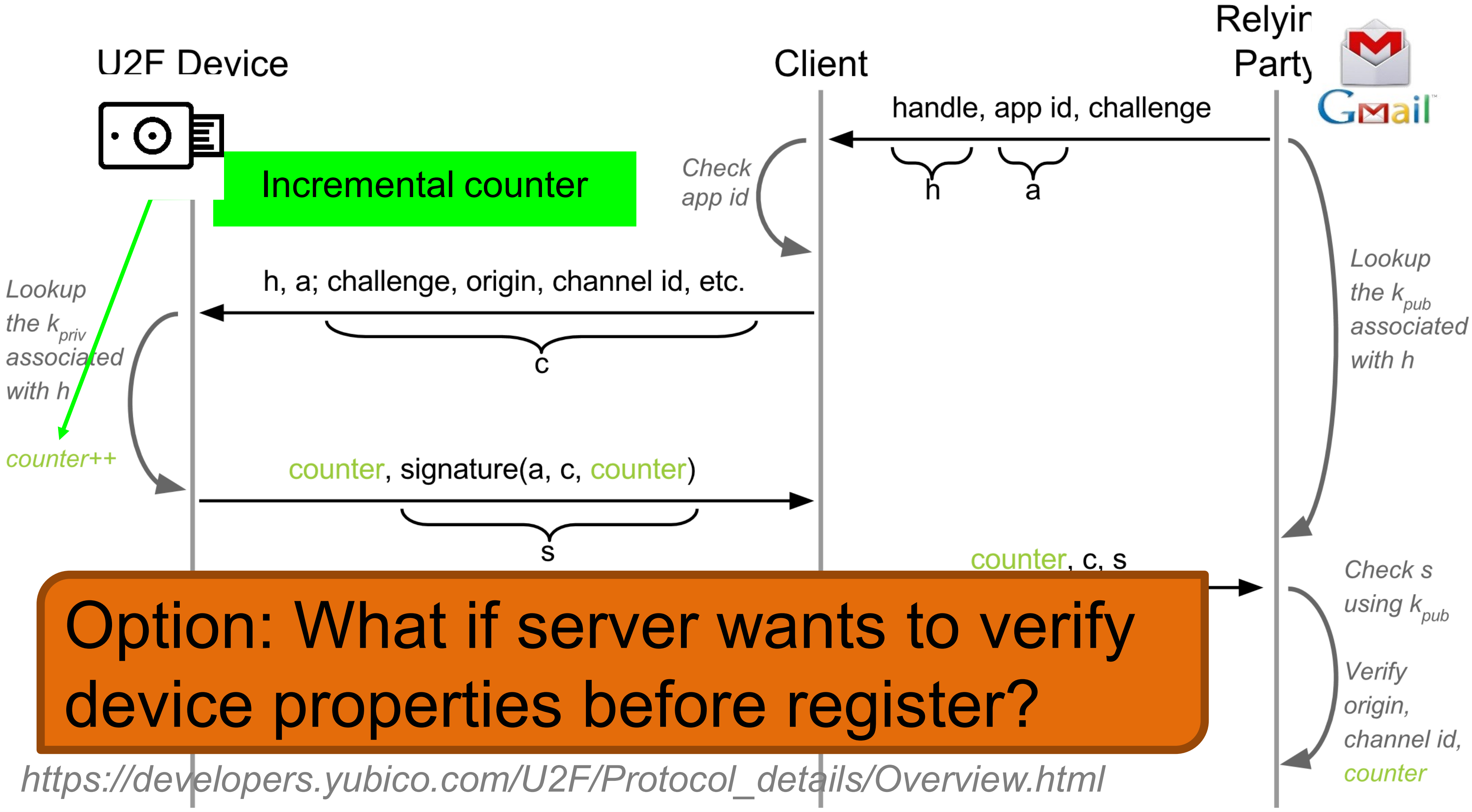

U2F Device
Client
Server
Gmail
handle, app id, challenge
h
a
heck
pp id
h, a; challenge, origin, channel id, etc.
c
Lookup the k_{priv} associated with h
signature(a,c)
s
c, s
Lookup the k_{pub} associated with h
Check s using k_{pub}
Verify origin and channel id

new key pair and key handle for each registration

Problem: Undetectable device cloning

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

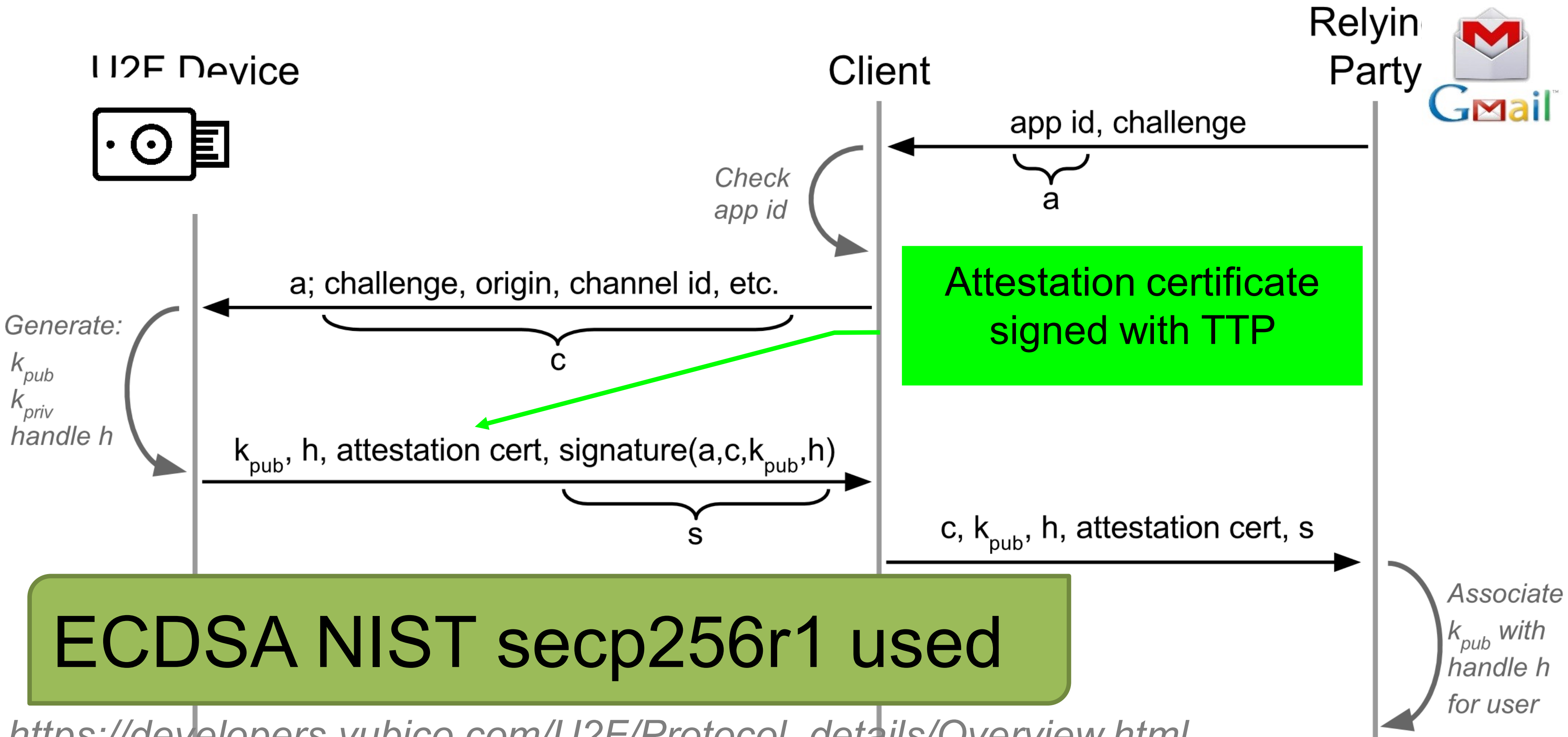
Revision 4: Authentication counter added



Option: What if server wants to verify device properties before register?

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

Revision 5: Device attestation added



Attestation certificate signed with TTP

ECDSA NIST secp256r1 used

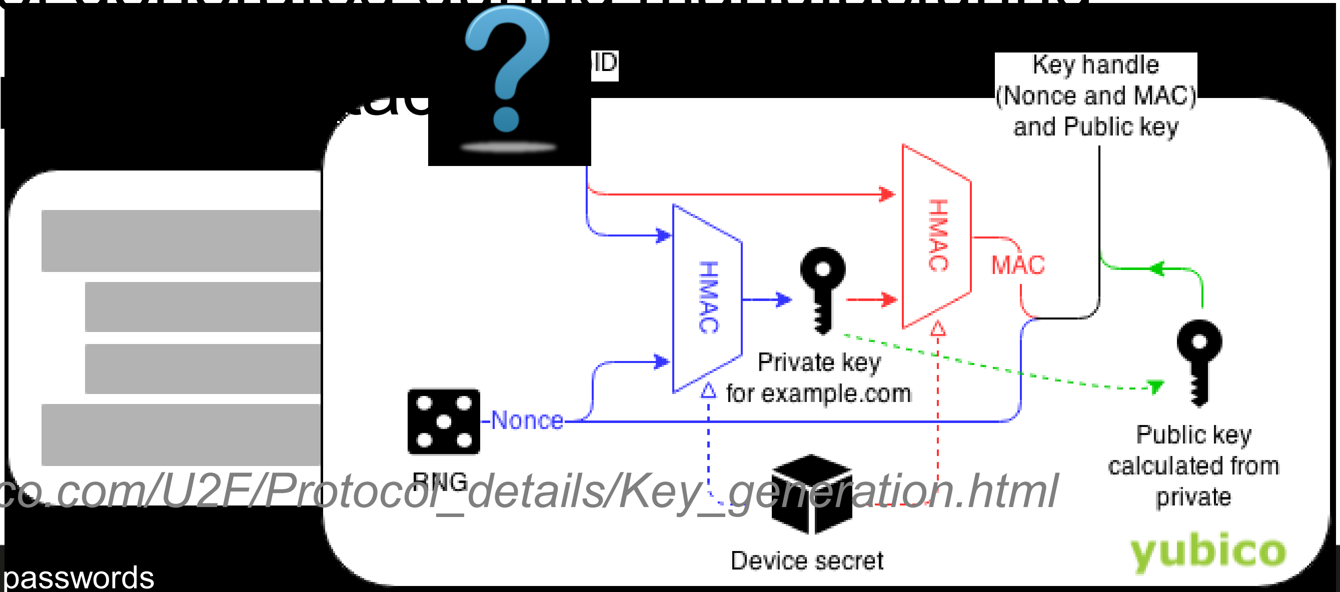
https://developers.yubico.com/U2F/Protocol_details/Overview.html

FIDO U2F devices

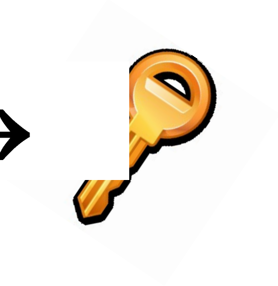
- Why have button? Is missing display problem?
- Recent problem: direct WebUSB API in Chrome
  - Malware bypass U2F API checking the URL
  - Legitimate URL is send from malicious page
  - https://www.wired.com/story/chrome-yubikey-phishing-webusb/
  - APDU-level communication: https://npmccallum.gitlab.io/post/u2f-protocol-overview/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

Always dig for implementation details

- How are ECC keys generated and stored?
- Yubikey saves ECC storage place by deriving ECC private keys instead of randomly generating
  - Possible as the ECC private key is random value
- Device secret generated during manufacturing
- What is the



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

H('Password') → 

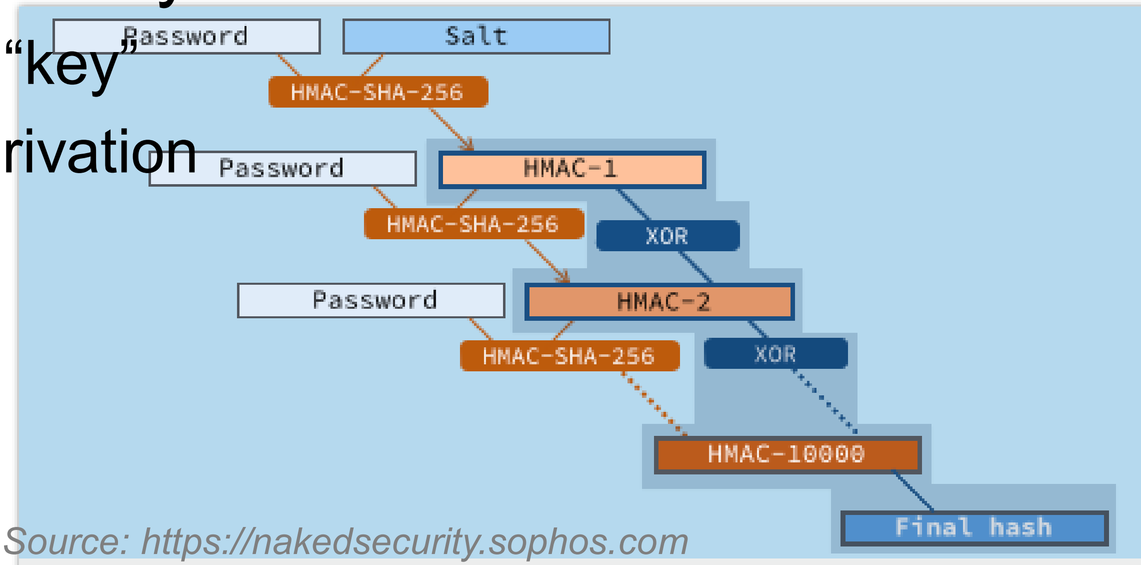
METHODS OF DERIVATION OF SECRETS FROM PASSWORD

Problems when password used as a key

- Passwords are usually shorter / longer than key
- If password as a key => low number of distinct keys
- Password does not contain same amount of entropy as binary key (only printable characters…)
- K = SHA-2(“password”)
  - Same passwords from multiple users => same key
  - Large pre-computed “rainbow” tables allow for quick check
  - Solved by addition of random (potentially public) *salt*
    - K = SHA-2(pass | salt)
- Dictionary-based brute-force still possible

Derivation of secrets from password

- PBKDF2 function, widely used
  - Password is HMAC "key"
  - Iterations to slow derivation
  - Salt added



Source: https://nakedsecurity.sophos.com

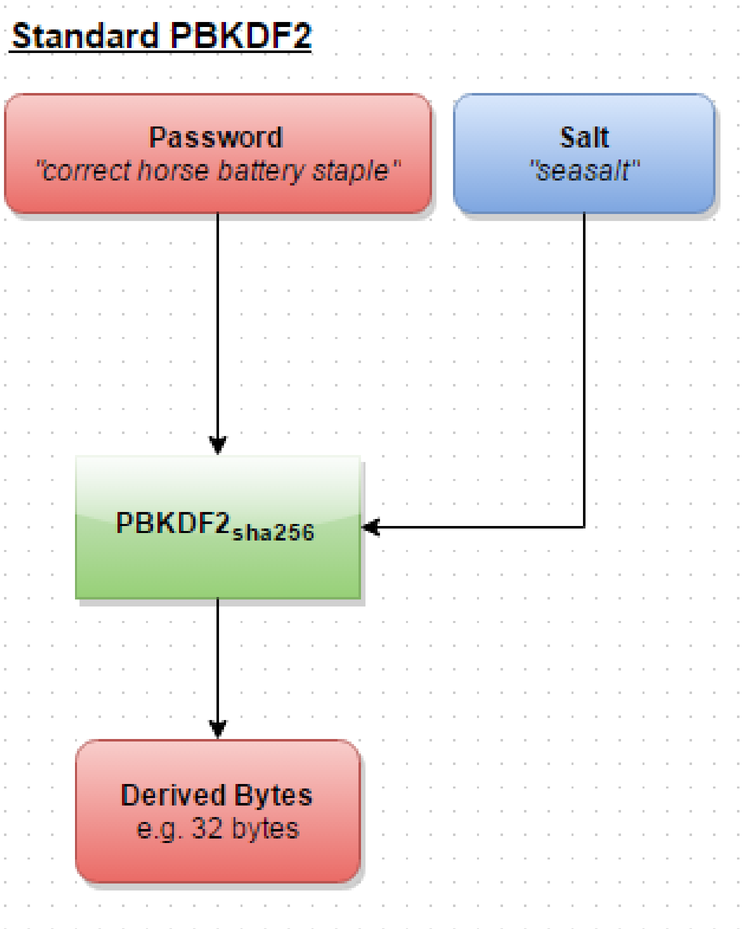
PBKDF2 using XOR to combine 10,000 successive HMAC-SHA-256 outputs into a final has

- Problem with custom-build hardware (GPU, ASIC)
  - Repeated iterations not enough to prevent bruteforce
  - (or would be too slow on standard CPU – user experience)

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)

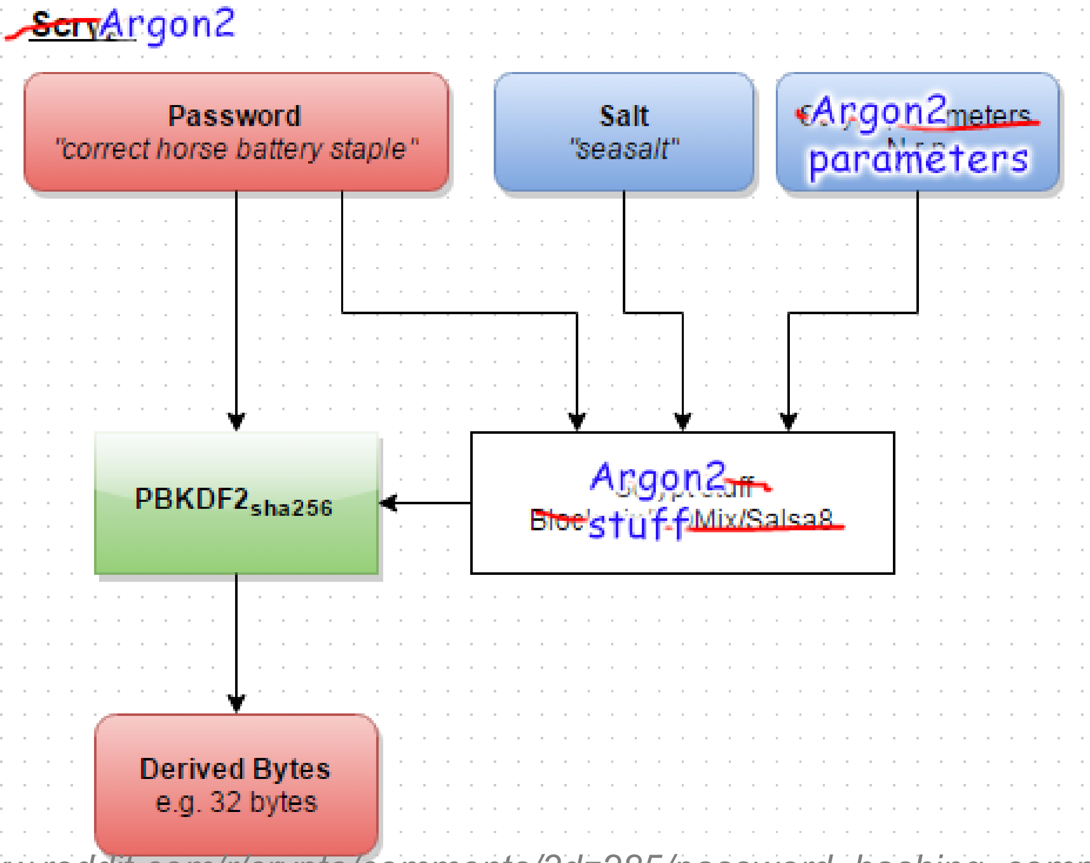
Reuse of external PBKDF2 structure



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

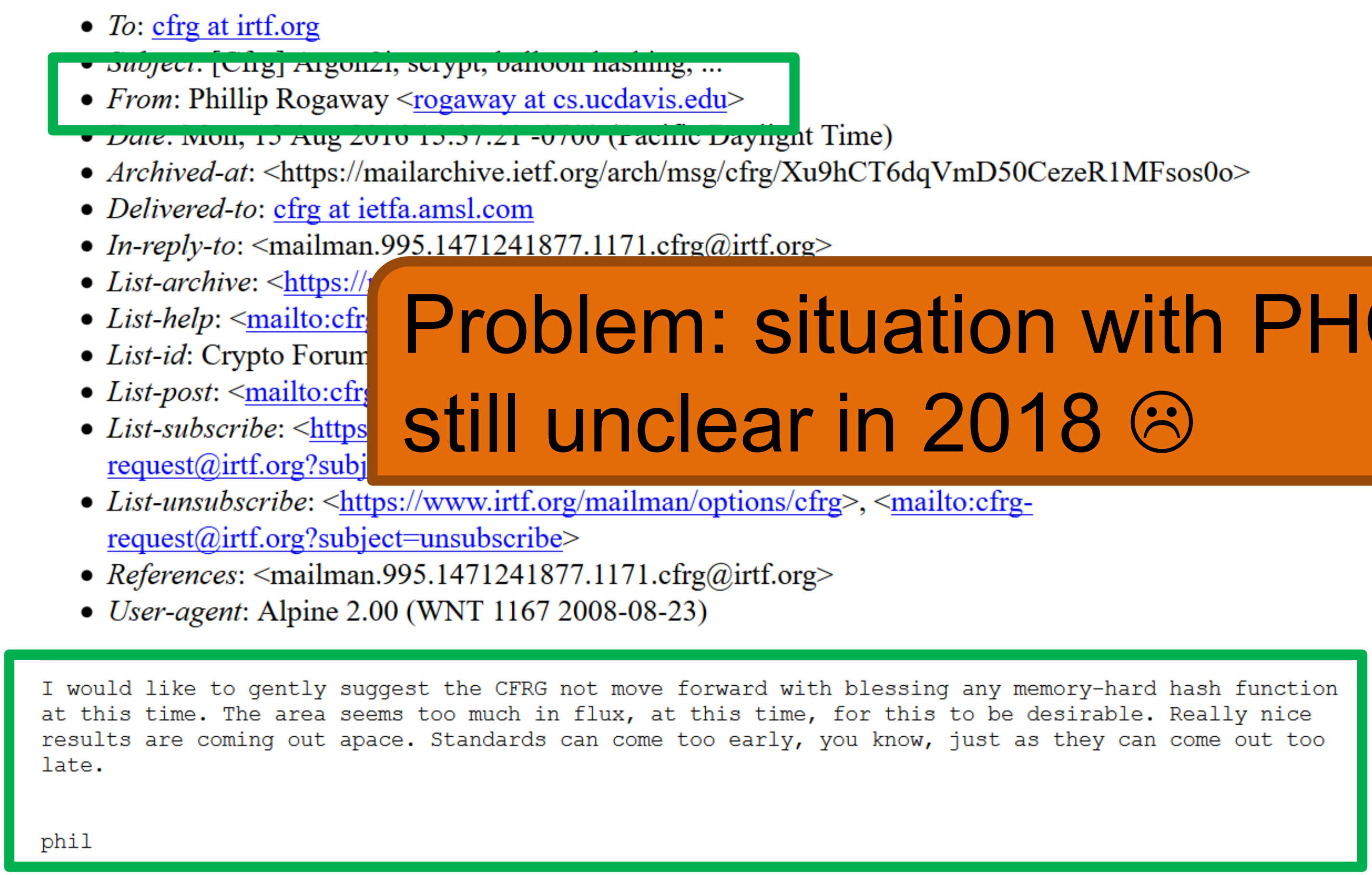
Argon2

- 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://
- *List-help*: <mailto:cfr
- *List-id*: Crypto Forum
- *List-post*: <mailto:cfr
- *List-subscribe*: <https
  request@irtf.org?subj
- *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
```

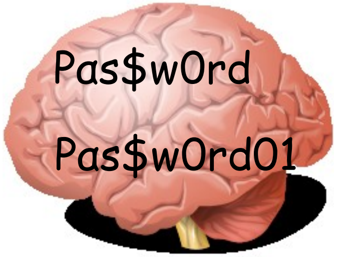
Problem: situation with PHC winner still unclear in 2018 ☹

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

PASSWORD MANAGERS

Evolution of password (managers)

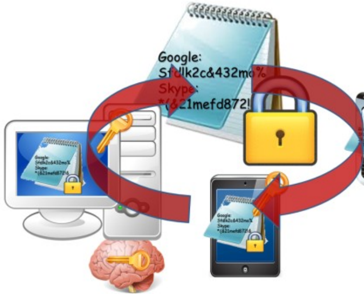
1. Human memory only



2. Write it down on paper

3. Write it into file

4. Use local password manager



Remote password managers

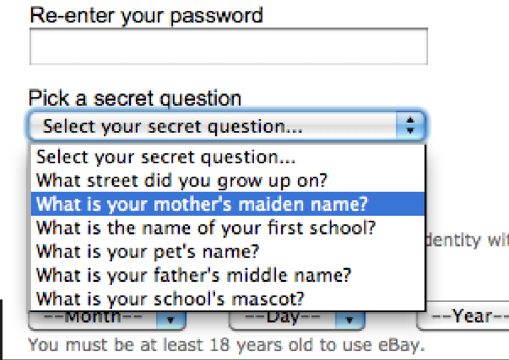
Google:
Sfdlk2c&432mo%
Skype:
*(&21mefd872!

KeePass+Dropbox
LastPass
1Password
MozillaSync
…

Common (mis-)Assumptions

1. User has strong password
   - >60% can be usually brute-forced

2. Server/service is hard to compromise
   - Server-side compromises are now very frequent
3. User have unique passwords
   - Gawker/root.com leak: 76% had the exact same password
4. Different authentication channels are independent
   - Web-browsing + SMS on smart phones?
5. Account recovery often weak(er)



Re-enter your password

Pick a secret question

Select your secret question...

Select your secret question...
What street did you grow up on?
What is your mother's maiden name?
What is the name of your first school?
What is your pet's name?
What is your father's middle name?
What is your school's mascot?

dentity wi

--Month-- --Day-- --Year--

You must be at least 18 years old to use eBay.

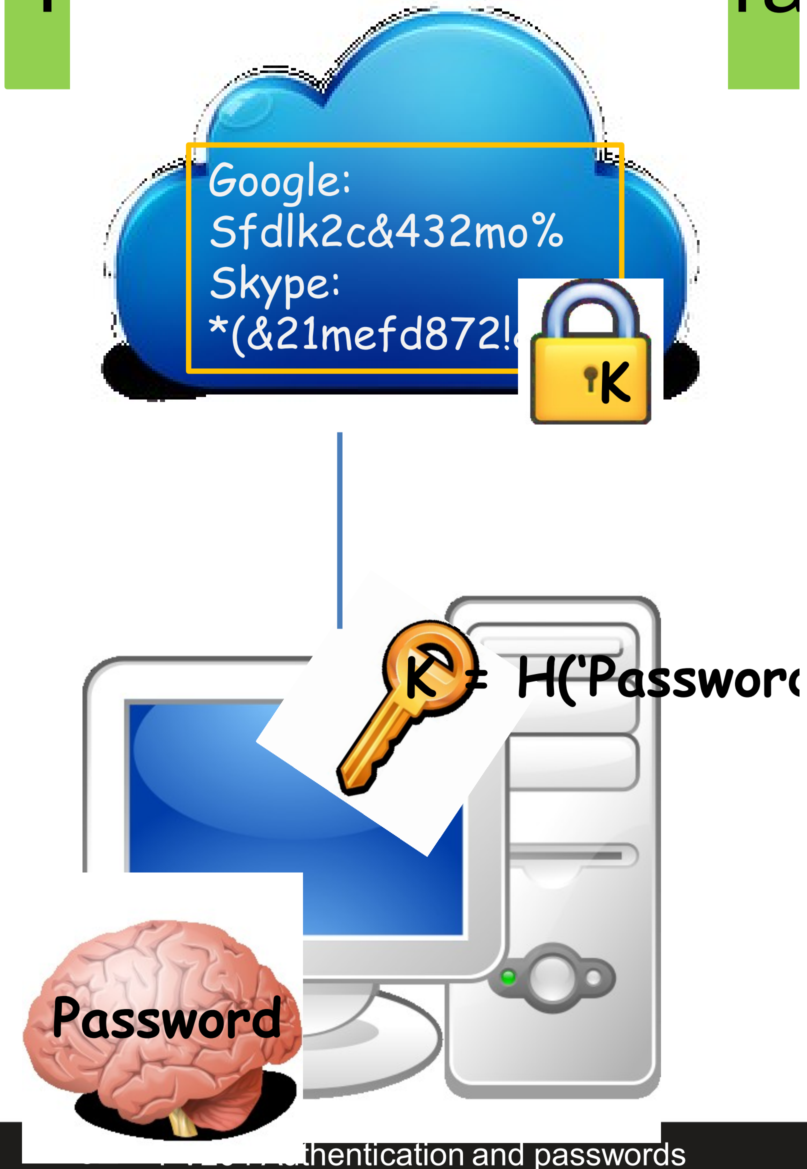
Case study

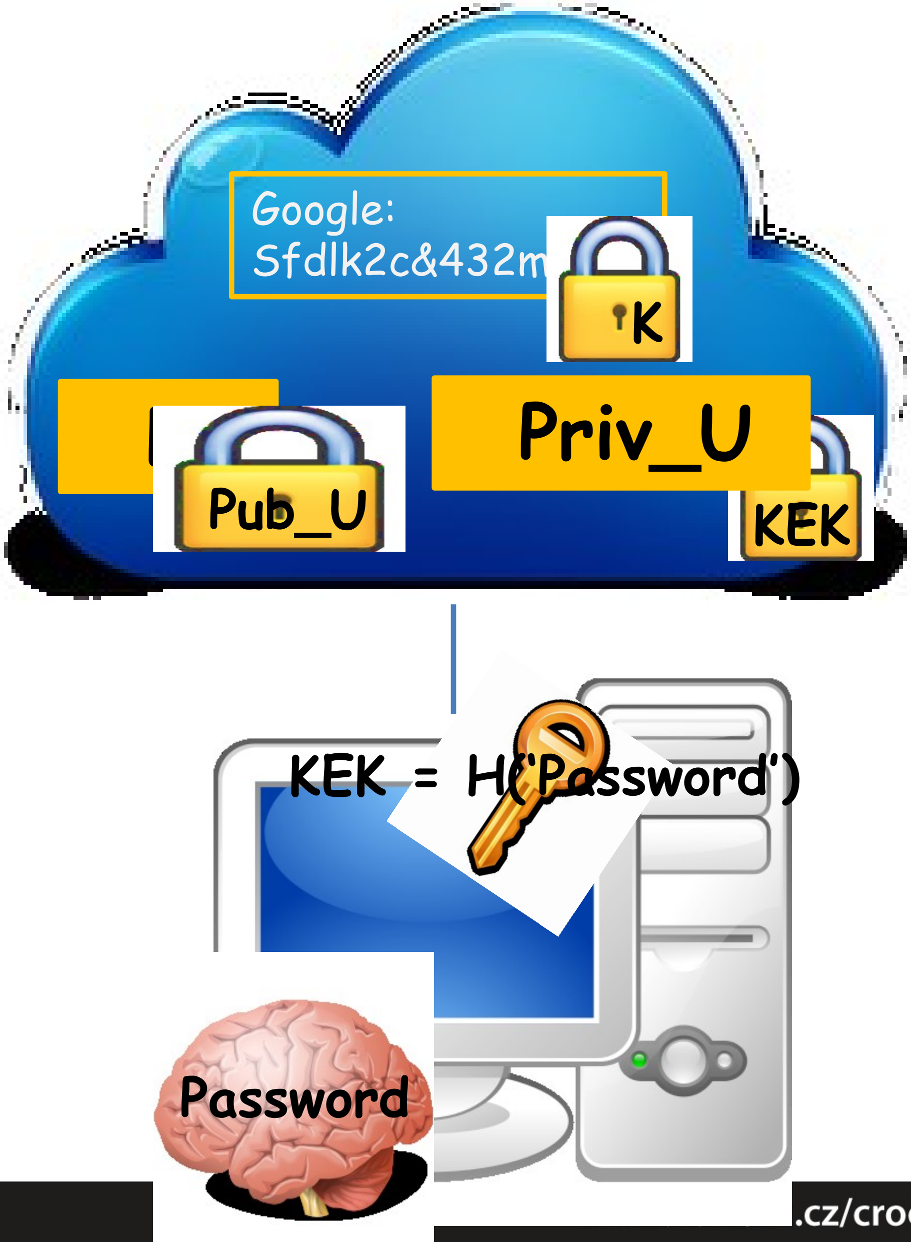
PASSWORD MANAGER FOR MULTIPLE DEVICES

Main security design principles

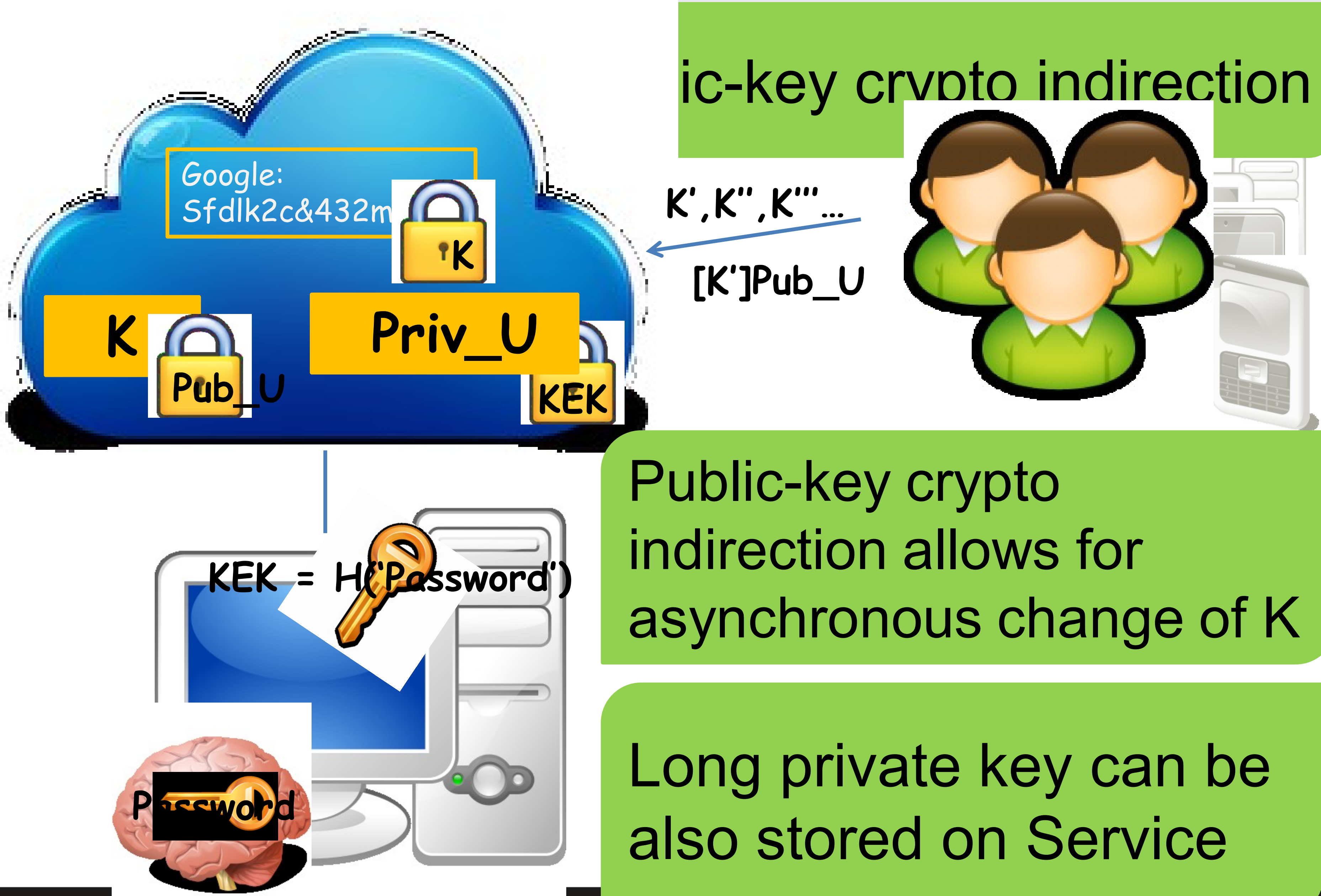
- Treat storage service as untrusted and perform security sensitive operations on client
- Make necessary trusted component as small as possible
- Prevent offline brute-force, but don’t expect strong password from user
  - add entropy from other source
- Make transmitted sensitive values short-lived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

F ra


Google:
Sfdlk2c&432mo%
Skype:
*(&21mefd872!
K
K = H('Passwor
Password

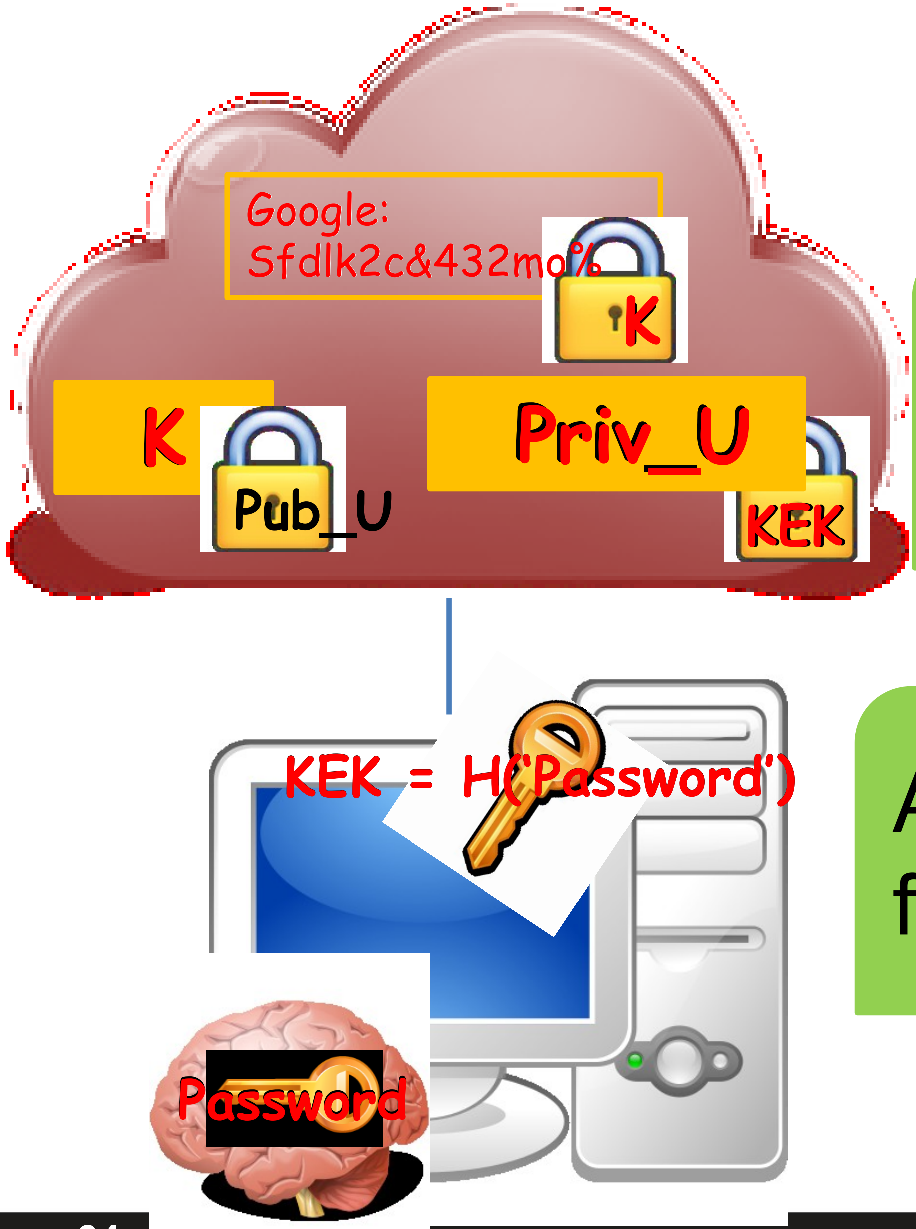

Google:
Sfdlk2c&432m
K
Priv_U
Pub_U
KEK
KEK = H('Password')
Password

ic-key crypto indirection



Public-key crypto indirection allows for asynchronous change of K

Long private key can be also stored on Service

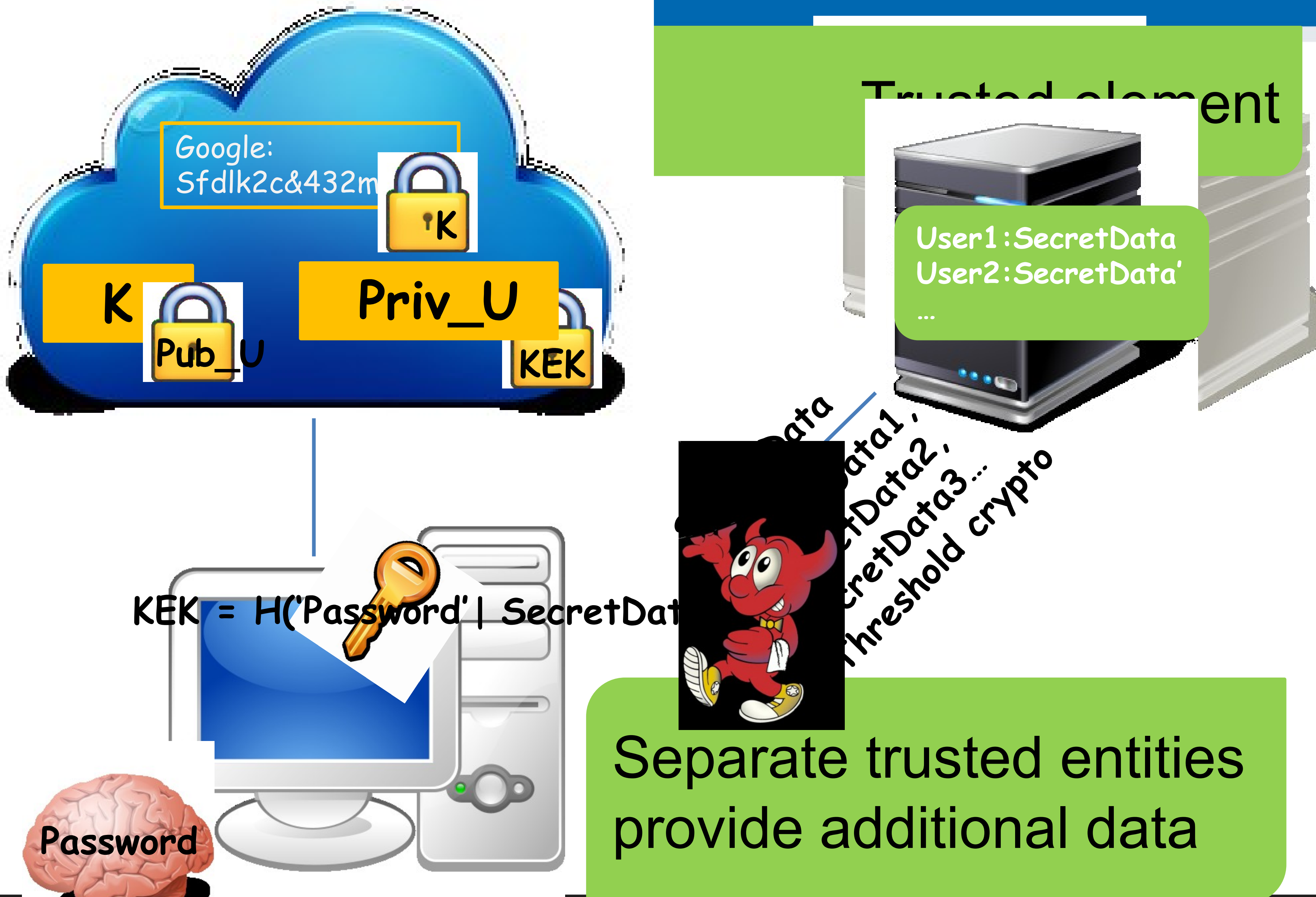


Weak password?

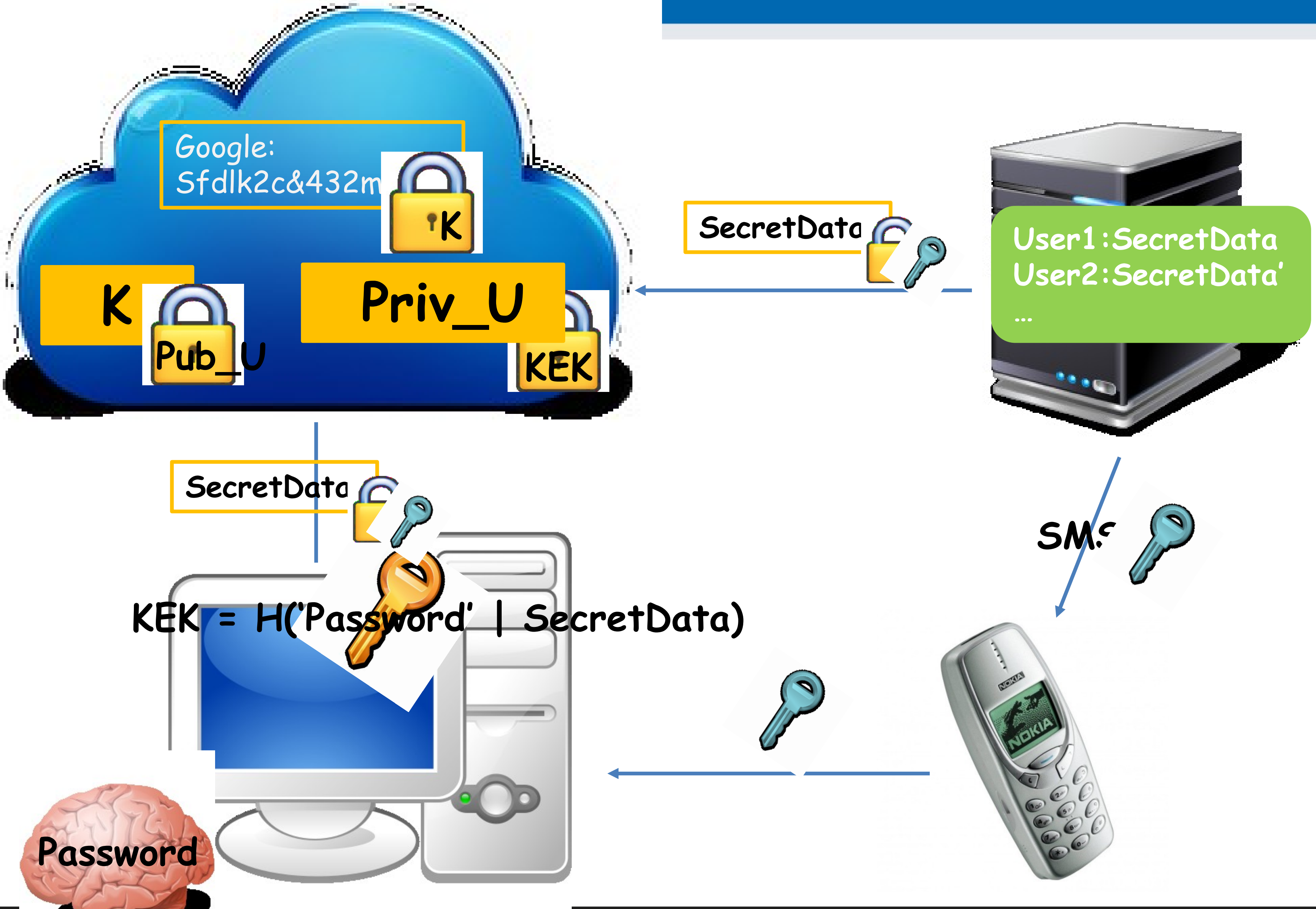
Users tend to have weak passwords…

Attacker has motivation for attacking the Service!

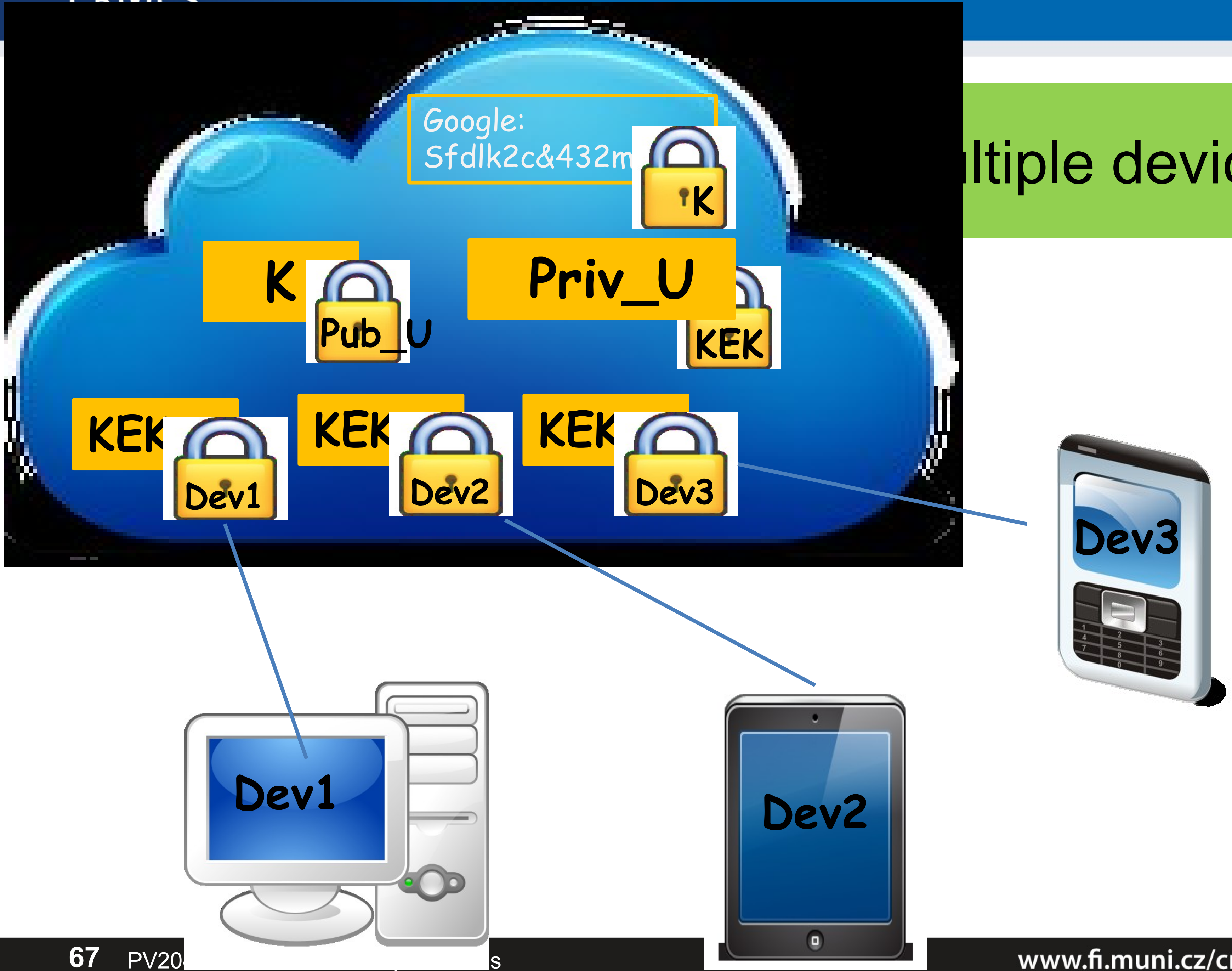
Trusted element



Separate trusted entities provide additional data


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

ltiple 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 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
  - 4 digits length is not an issue here – HSM enforce limited # retries

Summary

- Passwords have multiple issues, but are hard to be replaced
- Important to use passwords securely (guidelines)
- One-time passwords and tokens getting more used
- Password manager with synchronization over multiple devices is not straightforward
- Mandatory reading: UCAM-CL-817
  - At least chapters: II. Benefits, V. Discussion
  - Whole report is highly recommended