

PV204 Security technologies



Authentication and passwords

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

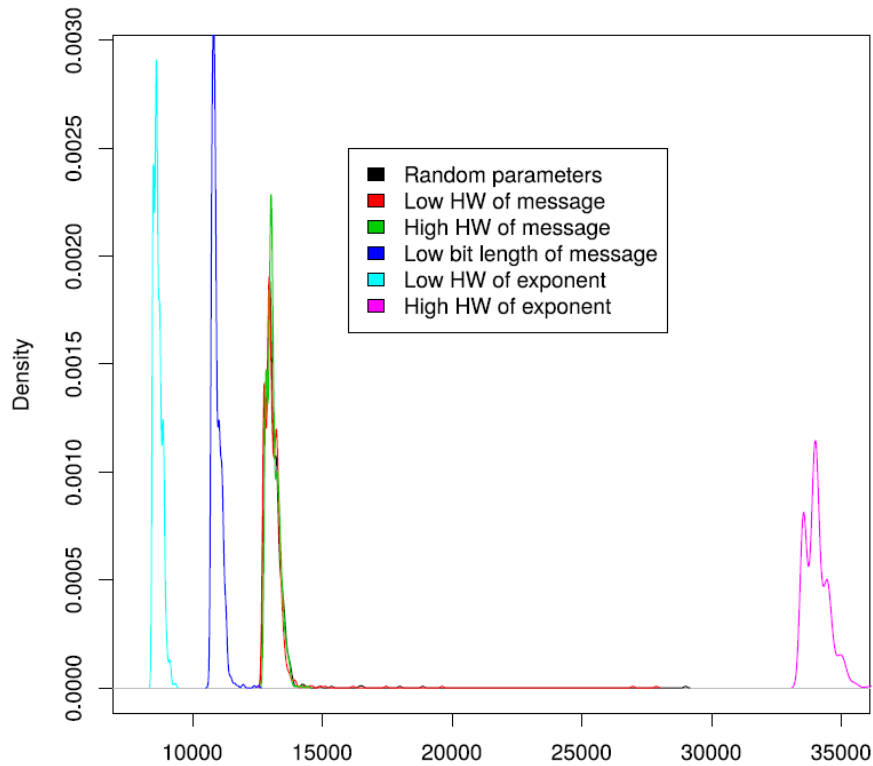


How to approach homework I.

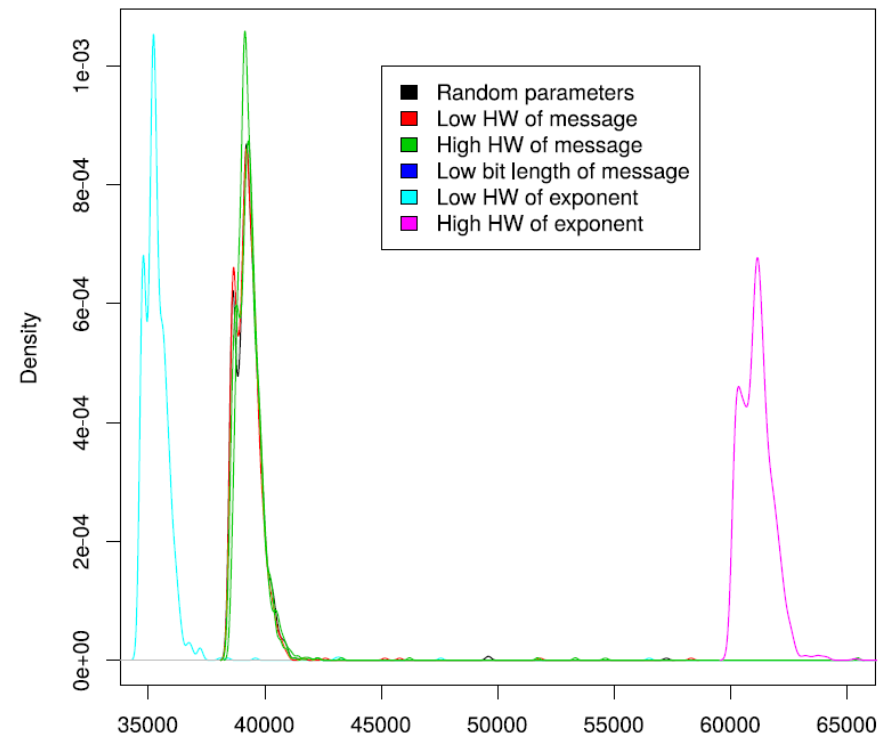
- What you should listed about your configuration (replication)
 - live vs. dedicated machine → impact on measurement
 - list platform configuration, compilation flags used (for replication)
- Analysis
 - good to print only shape of histogram instead of bars (visibility)
 - good to print multiple histograms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

Example solution (O. Mosnacek)

Simple square-and-multiply (squareAndMultiplyMPI)



Blinded version (squareAndMultiplyBlindedMPI)



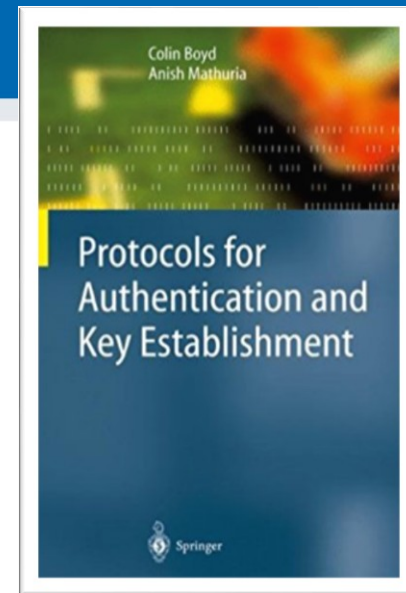
Conclusions for Homework I.

- Variability for same data and key \Rightarrow noise in measurement (\Rightarrow potentially harder to attack)
- Difference between any two measured values \Rightarrow possibility to use template attack
- Difference between data with low/mid/high hamming weight \Rightarrow some information is leaking
- Dependency of time on HW of private exponent may be possible to detect even for blinded RSA

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

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)`)

- Be aware of rainbow tables and brute-force crackers

3. Derive key: Password \rightarrow cryptographic key

- Example: `key = PBKDF2(password)`

4. Use to establish authenticated key

- Example: Password + Diffie-Hellman \rightarrow 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 target operation (e.g., authorization of request)
- ...

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 change after exposure

Password (hash) cracking

- Scenario: dump of passwords hash database
- Password cracking attacks
 - Brute-force attack (up to 7-8 characters)
 - Dictionary attack (inputs with higher probability)
 - 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 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)

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

OllyDbg - AES_PolarSSL.exe

File View Debug Plugins Options Window Help

CPU - main thread, module AES_Pola

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	75 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 800000	CMP DWORD PTR SS:[EBP-C],80
011D102C	74 14	JE SHORT AES_Pola.011D1042
011D102E	817D F4 C00000	CMP DWORD PTR SS:[EBP-C],0C0
011D1035	74 16	JE SHORT AES_Pola.011D104D
011D1037	817D F4 000100	CMP DWORD PTR SS:[EBP-C],100
011D103E	74 18	JE SHORT AES_Pola.011D1058
011D1040	EB 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	EB 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	EB 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	EB 0A	JMP SHORT AES_Pola.011D106D
011D1063	B8 00F8FFFF	MOV EAX,-800
011D1068	E9 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 000000	MOV DWORD PTR SS:[EBP-8],0
011D1086	EB 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

Dump - 0020B000..0020FFFF

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	.0000000000000000
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	@SecurePassword:
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	0*r#0...0/#
0020FA1F	01 60 FA 20 00 05 32 1D 01 01 00 00 00 00 08 E1 4F	0*0 #2#00...iB0
0020FA2F	00 F8 7C 4F 00 7C 75 74 0B 00 00 00 00 00 00 00	.0!0.lut.....
0020FA3F	00 00 E0 FD 7E 00 00 00 00 34 FA 20 00 58 01 00	..d2"....4..X0.
0020FA4F	00 9C FA 20 00 89 36 1D 01 34 CD 49 DA 00 00 00	.6. .e6#04=I r...
0020FA5F	00 6C FA 20 00 AA 33 68 76 00 E0 FD 7E AC FA 20	.l. .73hu.d2"%.
0020FA6F	00 F2 9E EE 76 00 E0 FD 7E 77 18 E1 52 00 00 00	..x-v.d2"wfBR...
0020FA7F	00 00 00 00 00 00 E0 FD 7E 00 00 00 00 00 00 00d2".....
0020FA8F	00 00 00 00 78 FA 20 00 00 00 00 00 00 FF FF FFx:.....
0020FA9F	FF 05 71 F2 76 EB 27 2C 24 00 00 00 00 C4 FA 20	'q=vU',\$.---
0020FAAF	00 C5 9E EE 76 6D 32 1D 01 00 E0 FD 7E 00 00 00	..x-vm2#0.d2"....

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

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

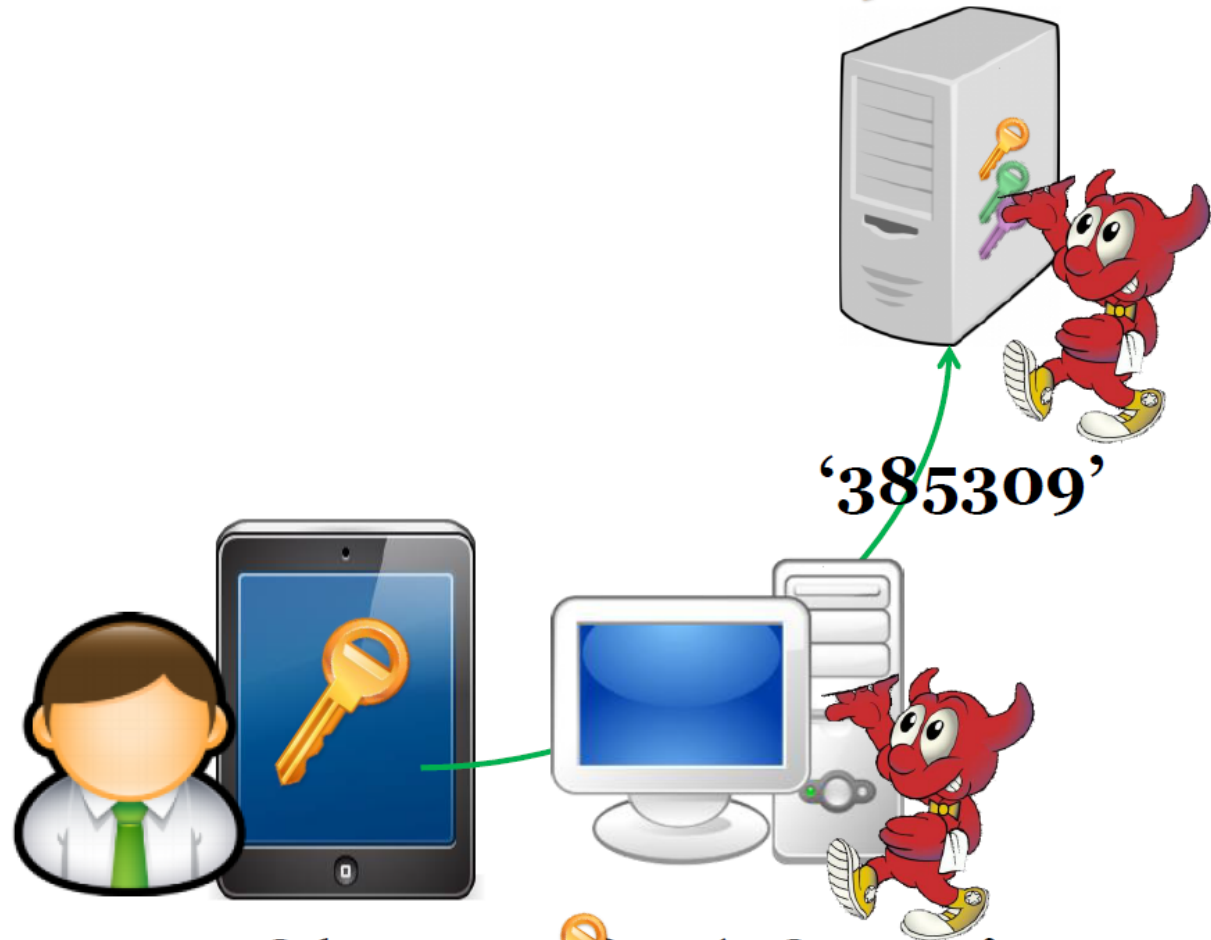
- HMAC-based One-time Password Algorithm (HOTP)
 - Secret key K
 - Counter (challenge) C
 - $HMAC(K, C) = SHA1(K \oplus 0x5c5c... \parallel SHA1(K \oplus 0x3636... \parallel C))$
 - $HOTP(K, C) = Truncate(HMAC(K, C)) \& 0x7FFFFFFF$
 - $0x7FFFFFFF$ mask to drop most significant bit (portability)
 - $HOTP\text{-Value} = HOTP(K, C) \bmod 10^d$ ($d \dots \#$ 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

Authentication server

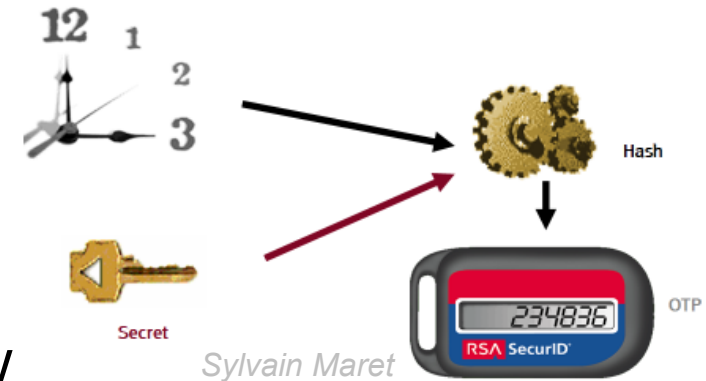
$\text{HMAC}(\text{ctr}++, \text{key}) == \text{'385309'}$?



$\text{HOTP} = \text{HMAC}(\text{ctr}++, \text{key}) = \text{'385309'}$

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

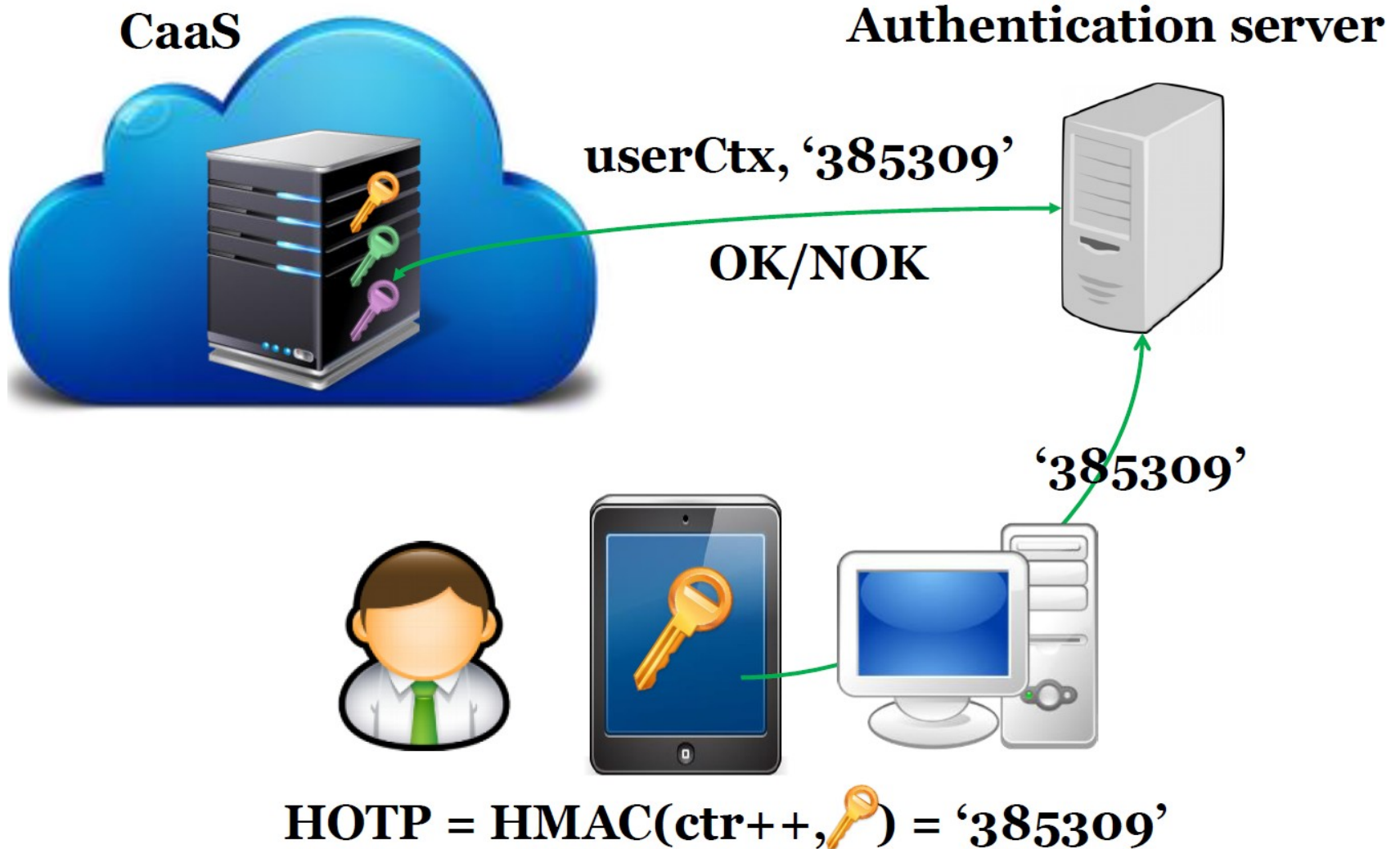


OCRA: OATH Challenge-Response Algorithm

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

Increased risk at *OTP verification server

- More secure against client compromise
 - Using OTP instead of passwords, $KDF(\text{time}|\text{key})$,
- 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



$H(\text{'Password'}) \rightarrow$ 

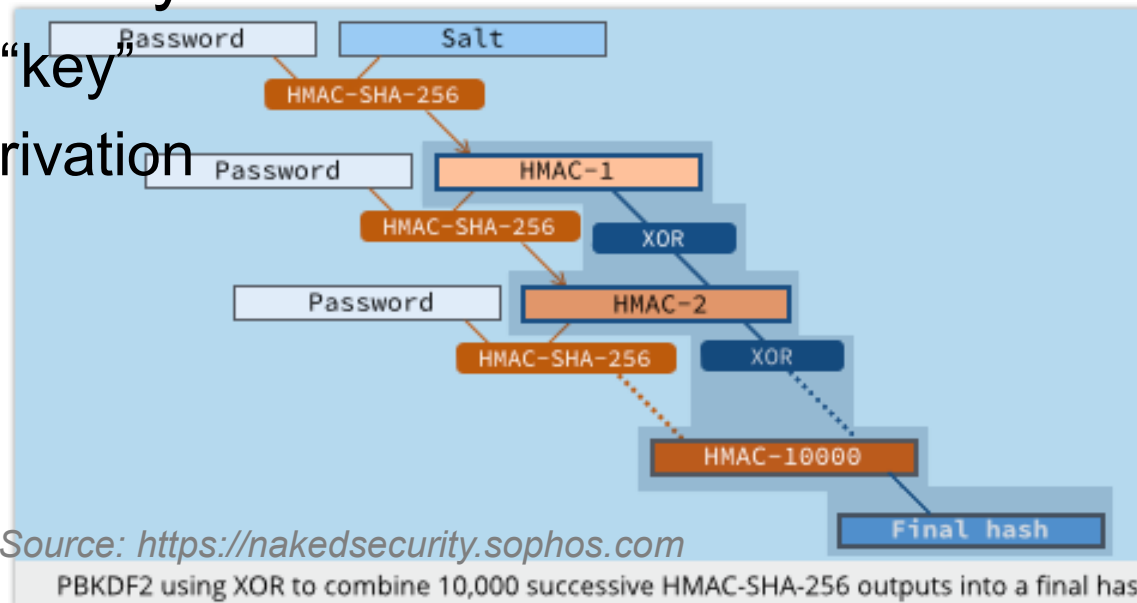
METHODS OF DERIVATION OF SECRETS FROM PASSWORD

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...)
- $K = \text{SHA-2}(\text{"password"})$
 - Same passwords from multiple users => same key
 - Large pre-computed “rainbow” tables allow for quick check
 - Solved by addition of random (potentially public) *salt*
 - $K = \text{SHA-2}(\text{pass} \mid \text{salt})$
- 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



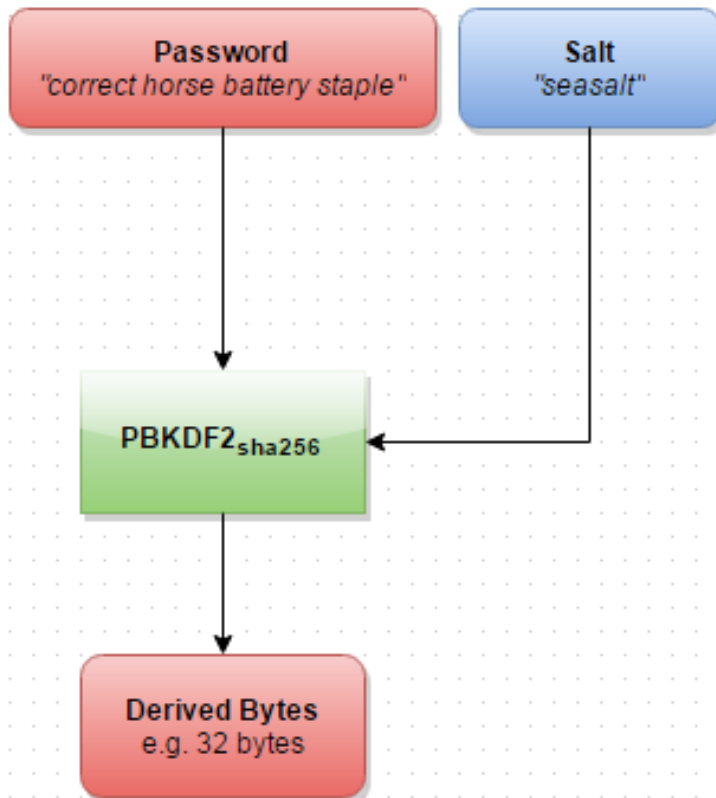
- Problem with custom-build hardware (GPU, ASIC)
 - Repeated iterations not enough to prevent bruteforce
 - (or would be too slow on standard CPU – user experience)

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

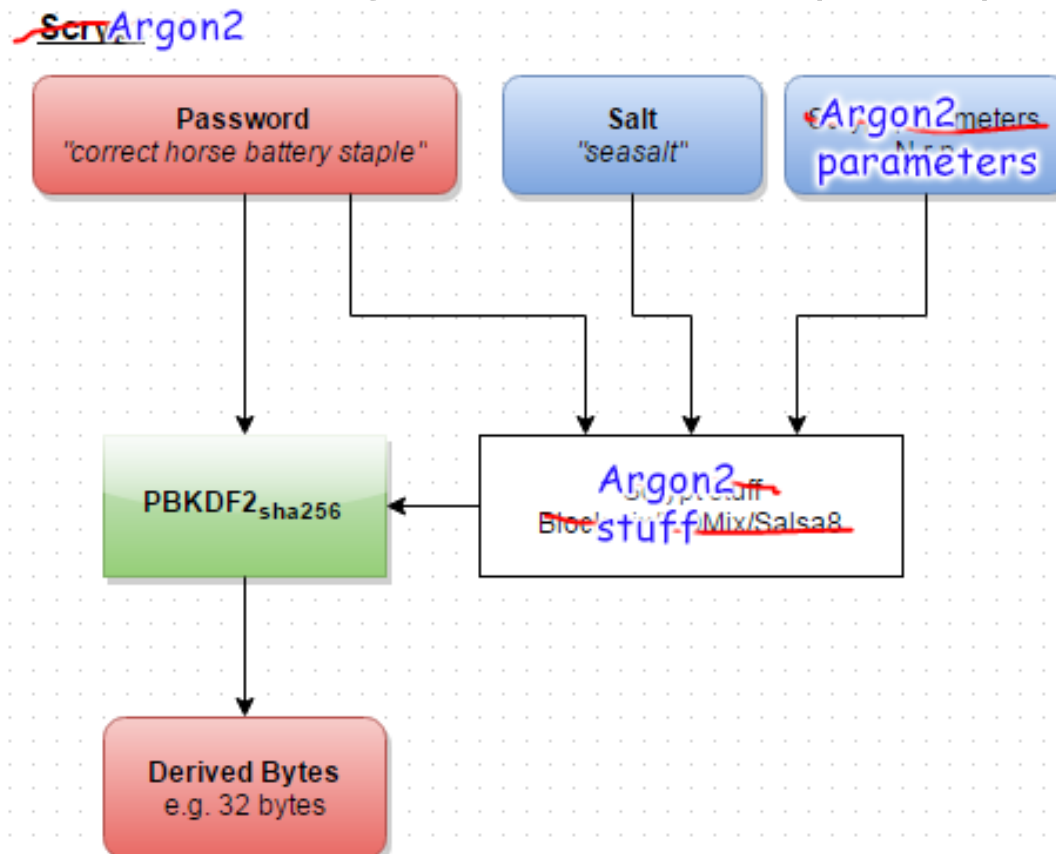
Standard PBKDF2



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](mailto:cfrg@irtf.org)
- Subject: [Cfrg] Argon2i, scrypt, balloon hashing, ...
- From: Phillip Rogaway <[rogaway at cs.ucdavis.edu](mailto:rogaway@cs.ucdavis.edu)>
- Date: Mon, 15 Aug 2016 15:57:21 -0700 (Pacific Daylight Time)
- Archived-at: <<https://mailarchive.ietf.org/arch/msg/cfrg/Xu9hCT6dqVmD50CezeR1MFsos0o>>
- Delivered-to: [cfrg at ietfa.amsl.com](mailto:cfrg@ietf.org)
- 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

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

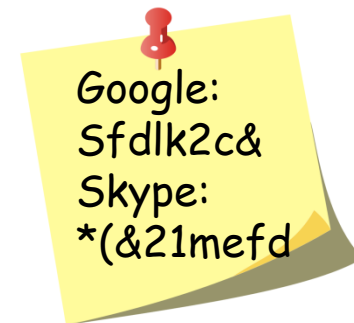
PASSWORD MANAGERS

Evolution of password (managers)

1. Human memory only



2. Write it down on paper



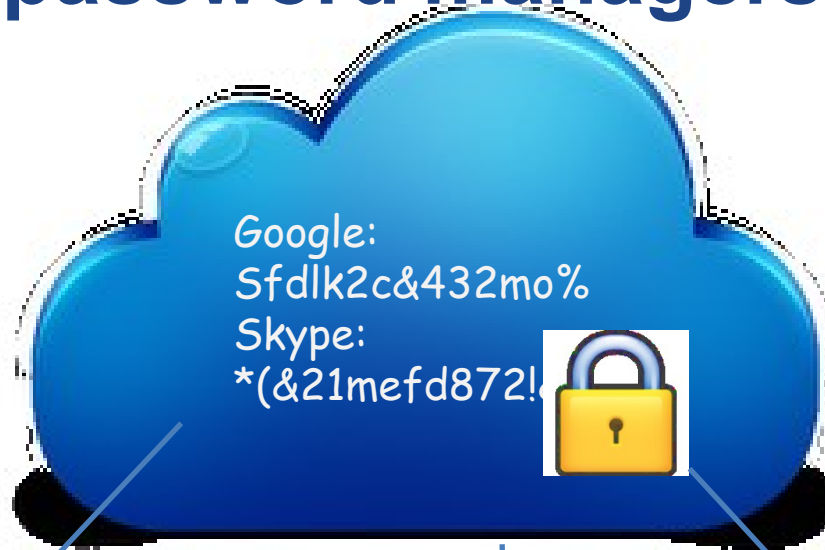
3. Write it into file



4. Use local password manager



Remote password managers



KeePass+Dropbox

LastPass

1Password

MozillaSync

....



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,



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 that he thinks it's more likely that a small number of user names and passwords



Human

k

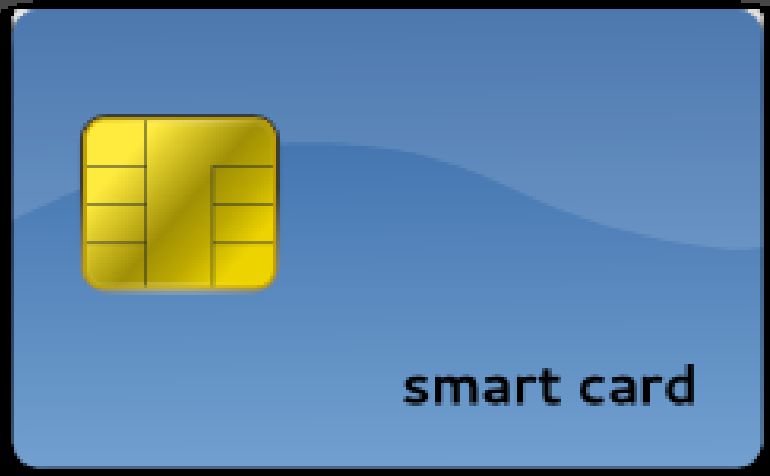


Google:
Sfdlk2c&432mo%
Skype:
*(&21mefd872!&



More than 60% of users have weak passwords

Hardware tokens



smart card



SmartArchitects' Password sCrib



Evergreen pas



Hack-a-Day's Mooltipass

Price, usability, compatibility...

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

User has strong password

nakedsecurity

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

malware mac facebook android vulnerability data loss privacy

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

LinkedIn confirms hack, over 60% of 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 passwords 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

TOP 30 PASSWORDS CRACKED



^ WORDS AMENDED FOR PUBLICATION. FEEL FREE TO USE YOUR IMAGINATION

User have unique passwords...

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

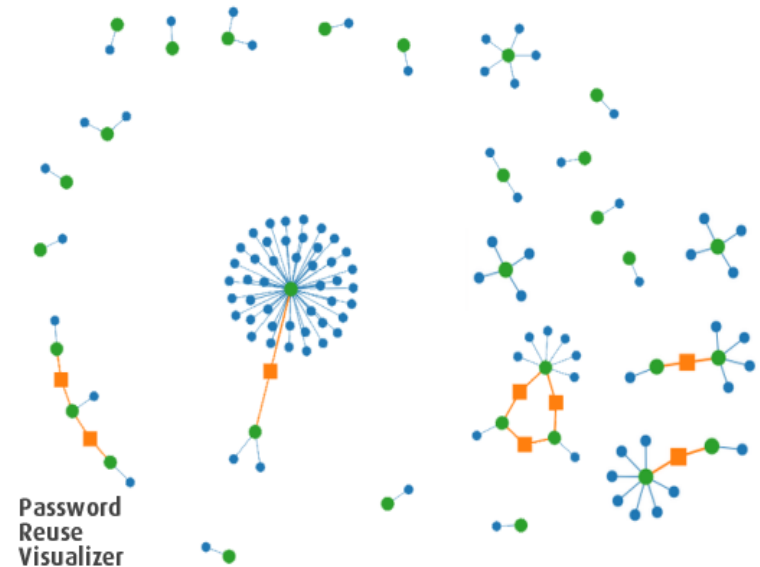
Follow @tsamson_IW

Print | 



More

Password Reuse Visualizer 0.13 by Paul Sawaya



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/>

problem of password reuse might be, and his conclusion is that it might be far worse than previous studies have indicated.

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 to own up to a password breach

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

eBay has admitted to a data breach of your password, now!



Scribd, "world's largest" admits to network intrusion breach

by Paul Ducklin on April 5, 2013 | 2 Comments



San Francisco-based document sharing site Scribd admitted to a network intrusion.

Details are scant, but fortunately the company suggests that most users are at risk...

Australian Defence Force Academy 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 possible 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 customers 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 that their usernames and passwords may have fallen into the 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

HOME MAIN MENU MY STORIES: 25 FORUMS SUBSCRIBE JOBS

RISK ASSESSMENT / SECURITY & HACKTIVISM

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 that they retrieved in plaintext from an unidentified service on Yahoo.

Service implementation is correct and free

PC

NEWS / OPINIONS / FEATURES / DEALS / HOW-TO / BUSINESS / SUBSCRIPTIONS

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 



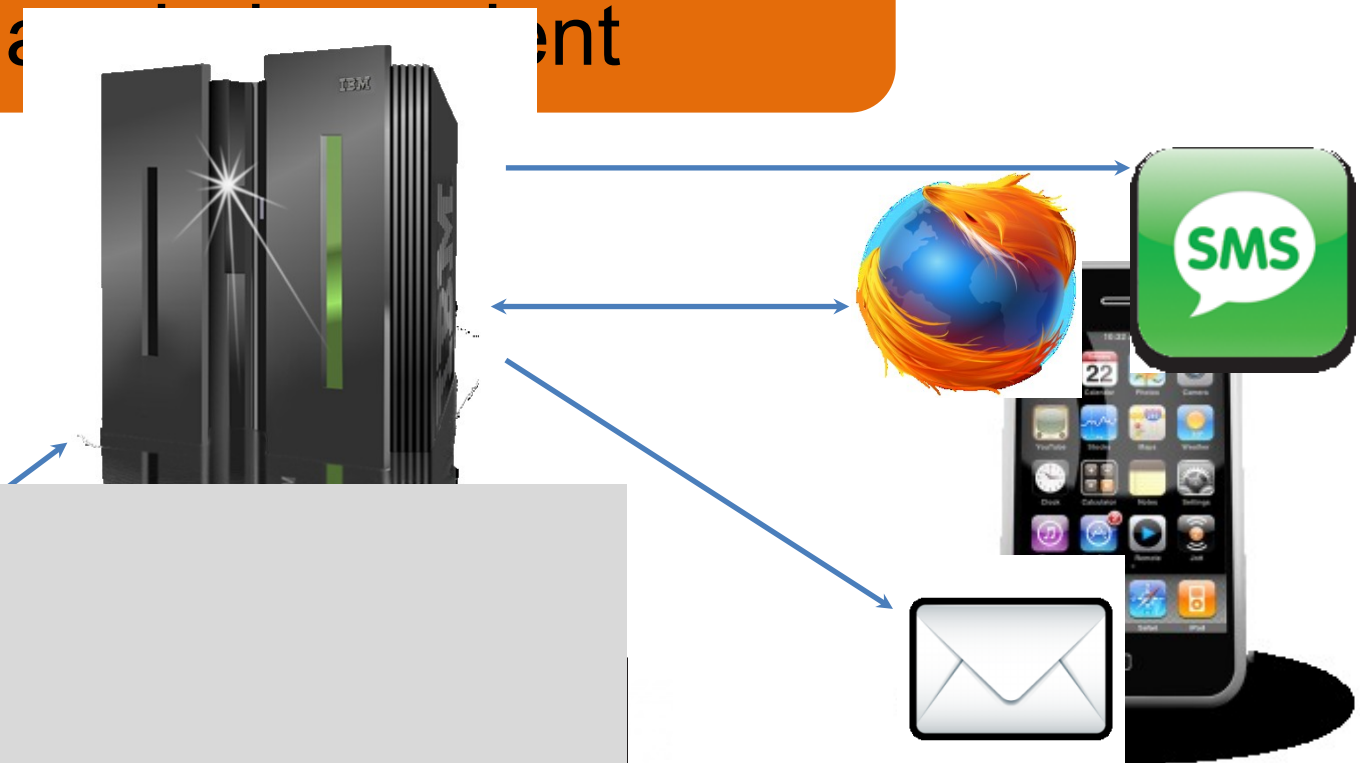
UPDATE: Starbucks on Friday released an updated version of its Starbucks Mobile App for iOS with additional performance safeguards, promising to protect mobile users. The update is available for download in the Apple App Store.

The Starbucks mobile app offers a lot of convenience when paying for coffee. However, it turns out, it could also be a security risk.

According to a report by The Verge, the massively popular international coffee chain has been storing



Different authentication channels and their implementation



Security can be maintained “forever”



Allow for some form of risk management

Access recovery is as secure as primary one

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?

ZDNet

White Papers Home

US Edition Mobility Research Windows

Topic: Security

Study: password recovery questions' easily guessed

Summary: How secret are in fact the 'secret questions' used for password recovery? Not so secret after all, according to a just published study by researchers at the University of North Carolina. The study's participants were not only able to answer the questions, but they also found that the most popular questions were in fact the easiest to guess.

ars technica

MAIN MENU MY STORIES: 16 FORUMS SUBSCRIBE JOBS

RISK ASSESSMENT / SECURITY & HACKTIVISM

How I lost my \$50,000 Twitter username

Stealing info from Pavpal & GoDaddy took just a minor bit of social engineering.

Recovery info shared over multiple services...

Case study

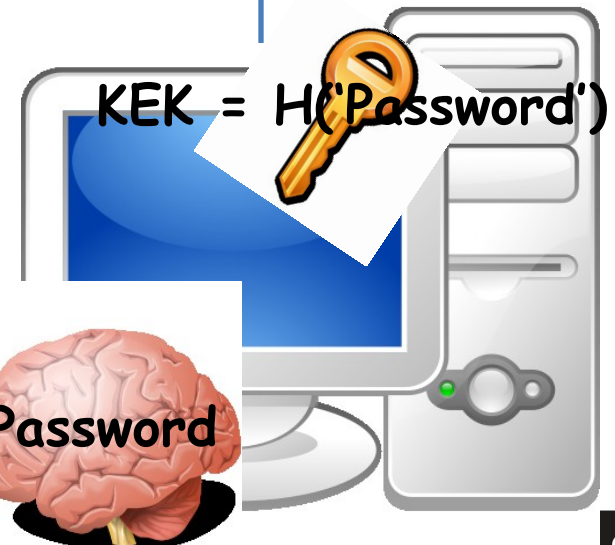
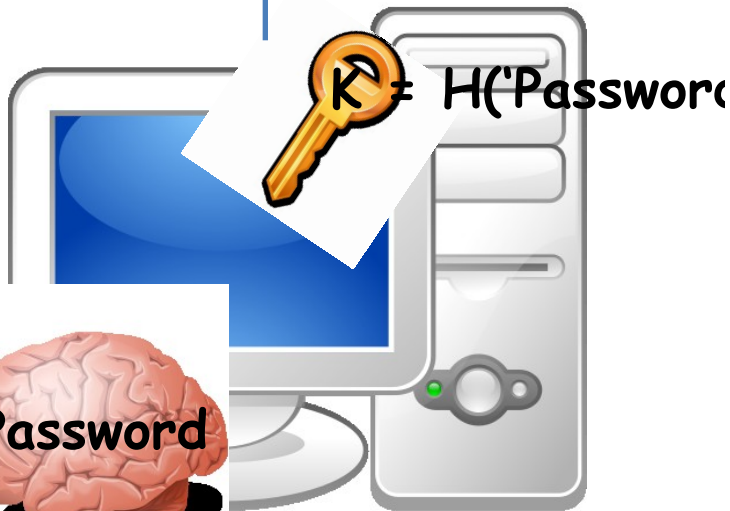
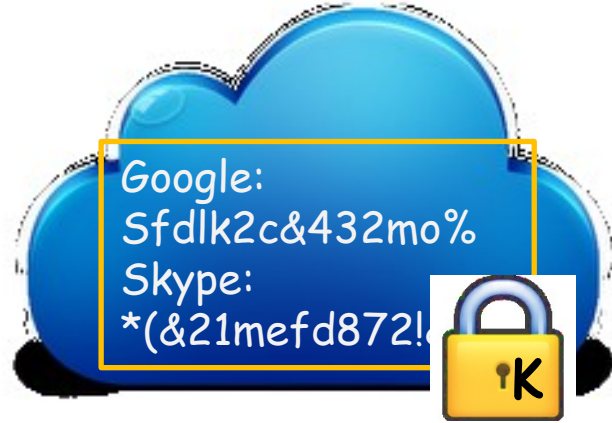
PASSWORD MANAGER FOR MULTIPLE DEVICES

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





Public-key crypto indirection

$K', K'', K''' \dots$
 $[K']_{Pub_U}$



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!





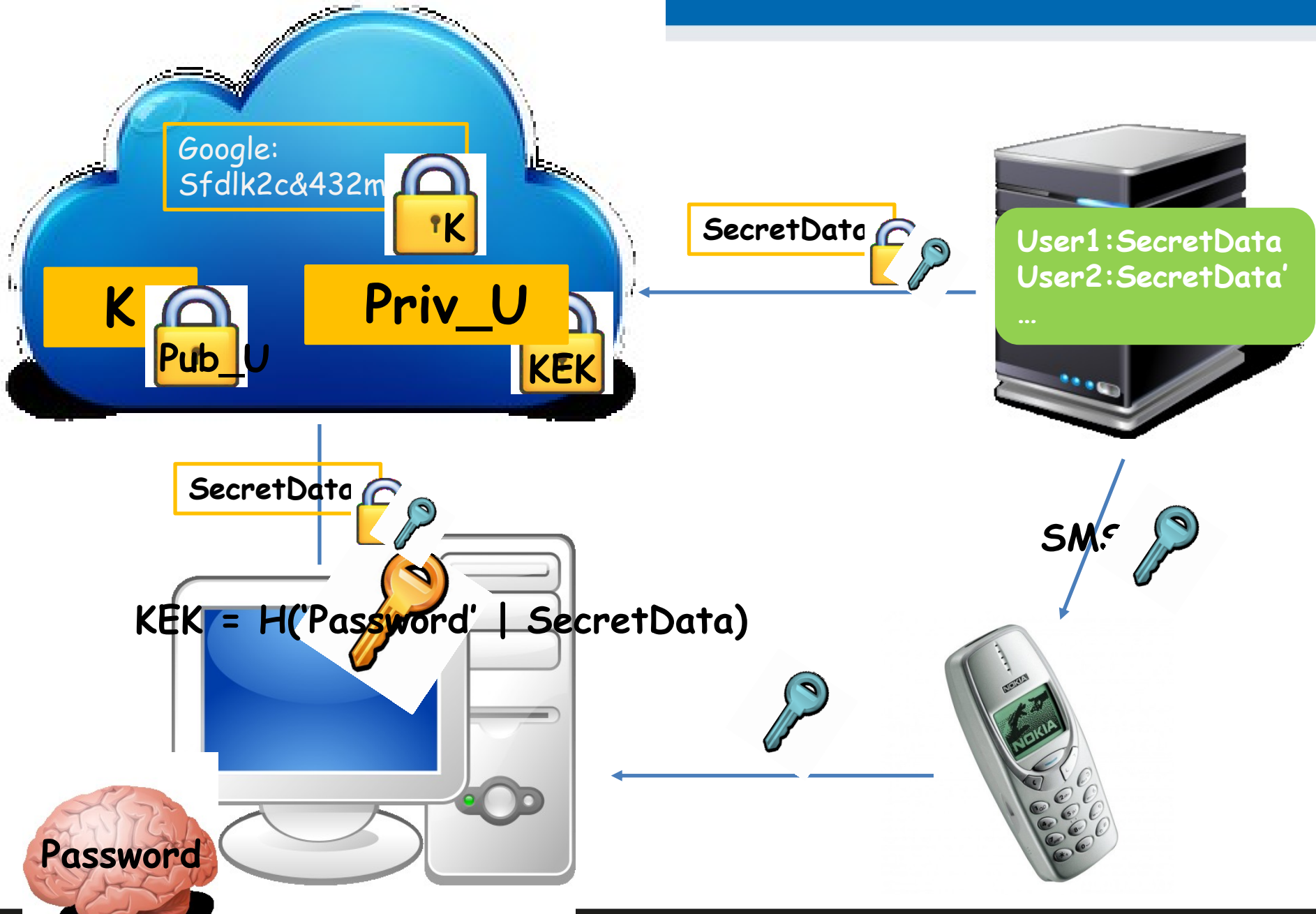
Trusted element



SecretData1,
SecretData2,
SecretData3...
threshold crypto

Separate trusted entities provide additional data





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



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 (recall recent FBI 10 password attempts)
- 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 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