

CSE508 Network Security

10/5/2017 **Authentication**

Michalis Polychronakis
Stony Brook University

Authentication

The process of reliably verifying the identity or role of someone (or something)

What is identity?

Which characteristics uniquely identify an entity?

Authentication is a critical service, as many other security mechanisms are based on it

Entity authentication is the security service that enables communicating parties to verify the identity of their peers

Two main types

Human to computer

Computer to computer



Credentials

Evidence used to prove an identity

User Authentication: credentials supplied by the user

- Something you know

- Something you have

- Something you are

Computer authentication: crypto, location

- Computers (in contrast to humans) can “remember” large secrets (keys) and perform complex cryptographic operations

- Location: evidence that an entity is at a specific place (e.g., IP address/subnet)

Authentication can be delegated

- The verifying entity accepts that a trusted third party has already established authentication

Something You Know: Password-based Authentication

Passwords, passphrases, pins, key-phrases, access codes, ...

Say the magic word

Good passwords are easy to remember and hard to guess

Easy to remember → easy to guess

Hard to guess → hard to remember

Bad ideas: DOB, SSN, zip code, favorite team name, ...

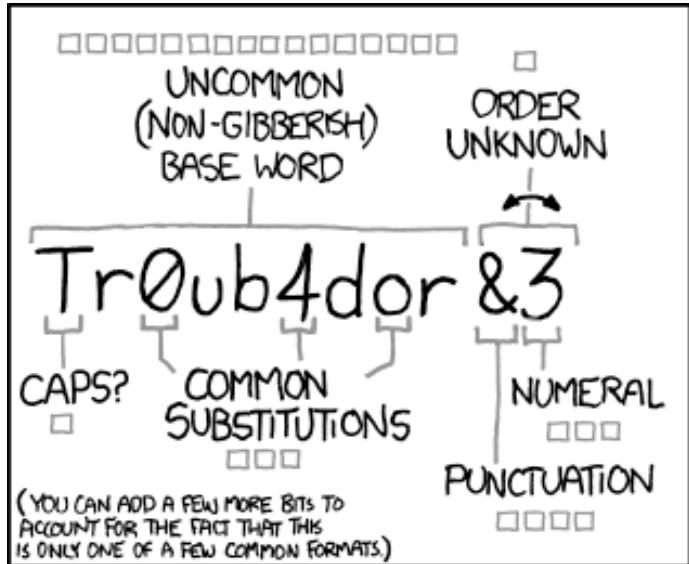
Password space (bits) depends on:

Password length

Character set

Better way to think about strong passwords

Long passphrases, combined with custom variations, symbols, numbers, capitalization, ...



~28 BITS OF ENTROPY

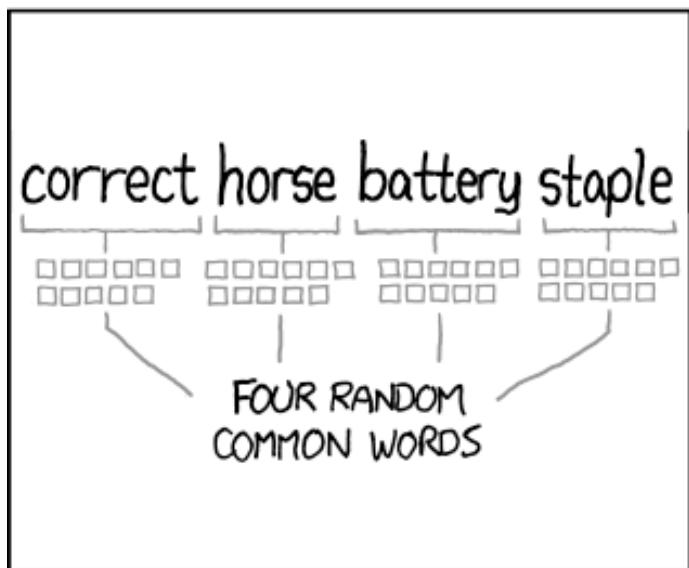
$2^{28} = 3 \text{ DAYS AT } 1000 \text{ GUESSES/SEC}$

(PLAUSIBLE ATTACK ON A WEAK REMOTE WEB SERVICE. YES, CRACKING A STOLEN HASH IS FASTER, BUT IT'S NOT WHAT THE AVERAGE USER SHOULD WORRY ABOUT.)

DIFFICULTY TO GUESS:
EASY

WAS IT TROMBONE? NO, TROUBADOR. AND ONE OF THE 0s WAS A ZERO?
AND THERE WAS SOME SYMBOL...

DIFFICULTY TO REMEMBER:
HARD



~44 BITS OF ENTROPY

$2^{44} = 550 \text{ YEARS AT } 1000 \text{ GUESSES/SEC}$

DIFFICULTY TO GUESS:
HARD

THAT'S A BATTERY STAPLE.
CORRECT!

DIFFICULTY TO REMEMBER:
YOU'VE ALREADY MEMORIZED IT

THROUGH 20 YEARS OF EFFORT, WE'VE SUCCESSFULLY TRAINED EVERYONE TO USE PASSWORDS THAT ARE HARD FOR HUMANS TO REMEMBER, BUT EASY FOR COMPUTERS TO GUESS.

Password Policies (often have the opposite effect)

Password rules

“At least one special character”

“Minimum/Maximum length of 8/12 characters”

“Must contain at least one number”

“Must contain at least one capital letter”

Hard to remember! → encourage password reuse, writing down passwords insecurely, ...

Periodic password changing

“You haven’t changed your password in the last 90 days”

Probably too late anyway if password has been stolen

Makes remembering passwords harder → more passwords resets

What users do: password1 → password2 → password1 → ...

Attacking Passwords

- Offline cracking
 - Online guessing
 - Eavesdropping
 - Capturing
- 
- Brute force attacks*

Password Storage

Storing passwords as plaintext is disastrous

Better way: store a cryptographic hash of the password 

Even better: store a “salted” version of the password

Defend against *dictionary attacks*: prevent precomputation of hash values (wordlists of popular passwords, rainbow tables, ...)

Even if two users have the same password, their hash values will be different → need to be cracked separately

Salting *does not* make brute-force guessing a given password harder!

| Username | Salt | Password hash  |
|----------|------|---|
| Bobbie | 4238 | h(4238, \$uperman) |
| Tony | 2918 | h(2918, 63%TaeFF) |
| Mitsos | 6902 | h(6902, zour1da) |
| Mark | 1694 | h(1694, Rockybrook#1) |

Still, password databases are getting leaked...

Password Cracking

Exhaustive search → infeasible for large password spaces

Dictionary attacks

- Language words

- Lists of previously leaked real user passwords

Variations, common patterns, structure rules

- Prepend/append symbols/numbers/dates, weird capitalization, 133tspeak, visually similar characters, intended misspellings, ...

Target-specific information

- DOB, family names, favorite team, pets, hobbies, anniversaries, language, slang, ...

- Easy to acquire from social networking services and other public sites

Particularly effective against “security questions”

Advanced techniques

- Probabilistic context-free grammars, Markov models, ...

Combination of all the above



Example hashes

If you get a "line length exception" error in hashcat, it is often because the hash mode that you have requested does not match the hash. To verify, you can test your commands against example hashes.

Unless otherwise noted, the password for all example hashes is **hashcat**.

Table of Contents ▾

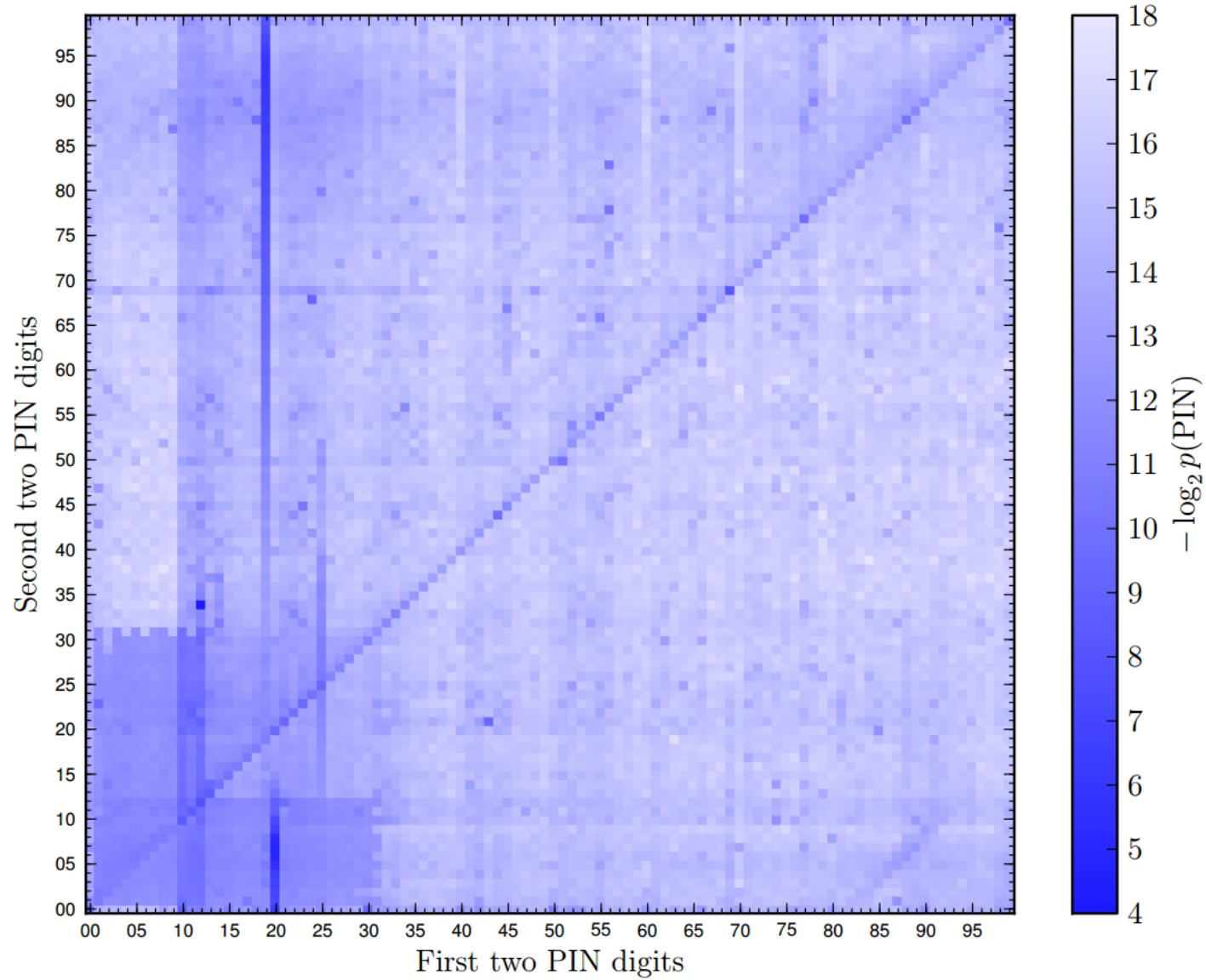
- Example hashes
- Generic hash types
- Specific hash types
- Legacy hash types

Generic hash types

| Hash-Mode | Hash-Name | Example |
|-----------|------------------------------|---|
| 0 | MD5 | 8743b52063cd84097a65d1633f5c74f5 |
| 10 | md5(\$pass.\$salt) | 01dfaee6e5d4d90d9892622325959afbe:7050461 |
| 20 | md5(\$salt.\$pass) | f0fda58630310a6dd91a7d8f0a4ceda2:4225637426 |
| 30 | md5(utf16le(\$pass).\$salt) | b31d032cfdf47a399990a71e43c5d2a:144816 |
| 40 | md5(\$salt.utf16le(\$pass)) | d63d0e21fdc05f618d55ef306c54af82:13288442151473 |
| 50 | HMAC-MD5 (key = \$pass) | fc741db0a2968c39d9c2a5cc75b05370:1234 |
| 60 | HMAC-MD5 (key = \$salt) | bfd280436f45fa38eaacac3b00518f29:1234 |
| 100 | SHA1 | b89eaac7e61417341b710b727768294d0e6a277b |
| 110 | sha1(\$pass.\$salt) | 2fc5a684737ce1bf7b3b239df432416e0dd07357:2014 |
| 120 | sha1(\$salt.\$pass) | cac35ec206d868b7d7cb0b55f31d9425b075082b:5363620024 |
| 130 | sha1(utf16le(\$pass).\$salt) | c57f6ac1b71f45a07dbd91a59fa47c23abcd87c2:631225 |
| 140 | sha1(\$salt.utf16le(\$pass)) | 5db61e4cd8776c7969cf62456da639a4c87683a:8763434884872 |
| 150 | HMAC-SHA1 (key = \$pass) | c898896f3f70f61bc3fb19bef222aa860e5ea717:1234 |
| 160 | HMAC-SHA1 (key = \$salt) | d89c92b4400b15c39e462a8caa939ab40c3aaaa:1234 |
| 200 | MySQL323 | 7196759210defdc0 |
| 300 | MySQL4.1/MySQL5 | fcf7c1b8749cf99d88e5f34271d636178fb5d130 |

25 Most-used (Worse) Passwords

| | | |
|----------|----------|----------|
| password | letmein | 2000 |
| 123456 | monkey | jordan |
| 12345678 | 696969 | superman |
| 1234 | abc123 | harley |
| qwerty | mustang | 1234567 |
| 12345 | michael | |
| dragon | shadow | |
| pussy | master | |
| baseball | jennifer | |
| football | 111111 | |



Distribution of 4-digit sequences within RockYou passwords

Wordlists

| | | | | | | |
|----------------|--------------|---------------|---------------|---------------|---------------|---------------|
| ce#ebc.dk | 4637324 | gea8mw4yz | fujinshan | masich | gothpunksk8er | 20081010 |
| goddess5 | bugger825 | kukumbike | counter | pengaiwei | rftaeo48 | leelou44 |
| 20071002 | marmaris | 260888 | N8mr0n | coalesce | 8d7R0K | 8UfjeGb0 |
| 271075711 | jinjin111 | jordi10 | 520057 | 56402768 | 5172032 | 200358808 |
| zs3cu7za | 170383gp | lexusis | adc123 | thesis | aics07 | dellede |
| scoopn | 3484427 | kj011a039 | bmaster | aabbcc894 | 34mariah | liang123. |
| frygas1411 | f133321 | c84bw1rb | qbjh04zg | marion&maxime | dongqinwei | captainettekt |
| SL123456sl | zwqrfg | priyanka05 | ueldaa79 | 614850 | samarica | kwiki-mart |
| 12345687ee123 | 67070857 | loveneverdies | EMANUELLI | ydz220105 | cap1014 | mdovydas |
| xuexi2010 | 432106969 | u8Aqebj576 | yanjing | 584521584521 | 0167387943 | tigmys2001 |
| daigoro | 6856 | FGYfgy77 | assynt | txudecp | AE86Trueno | denial |
| 12345614 | 704870704870 | 659397 | 62157173 | 84410545 | 19700913 | 678ad5251 |
| DICK4080 | pv041886 | 327296 | 0704224950753 | pietro.chiara | mcsuap | woaiwuai |
| 567891234 | 20060814 | 74748585 | 6903293 | jman1514 | bu56mpbu | 1591591591212 |
| tilg80 | 512881535 | 19720919 | axaaxa | heryarma | danbee | hNbDGN |
| 6z08c861 | milanimilani | 050769585 | hilall | 39joinmam | passw<> | cardcap |
| :zark: | 472619 | nicopa | 30091983 | timelapse | money521 | 13985039393 |
| ravishsneha | dbyxw888 | 2232566 | 2510618981 | mwinkar | conan83 | 001104 |
| 150571611369 | 85717221 | bearss | soukuokpan | 251422 | nxfjpl | desare11 |
| 661189 | cc841215 | n0tpublic | tosecondlife | willrock | rateg143 | 412724198 |
| passme | ariana19321 | isitreal00 | p4os8m6q | YHrtfgDK | kojyihen | nibh1kab |
| trolovinasveta | bbbnnn | ashraf19760 | 015614117 | xys96exq | 058336257 | asfeng |
| abdulkhaleque | ang34hehiu | 48144 | acw71790 | mercadotecnia | sarah4444 | hqb555 |
| 007816 | wj112358 | 22471015 | lsyljm2 | 8s5sBEx7 | 7363437 | xgames7 |
| xLDSX | Brenda85 | antyzhou115 | 2xgialdl | 0125040344 | freindship | muckerlee |
| Florida2011 | 786525pb | 0167005246 | gaybar9 | margitka | JytmvW0848 | choqui67 |
| 037037 | shi461988 | ec13kag | 88203009 | omaopa | sb inbau | 12130911 |
| WestC0untry | pingu | 226226226226 | MKltyh87 | dfTi6nh | 30907891 | lierwei120 |
| hitsugaiya | yebozip | 6767537/33 | quiggle | 1314520521 | 0515043111 | skytdvn |
| 955998126 | 71477nak | mimilebrock | 2063775206 | pixma760 | 1973@ati | milena1995 |
| 3n3rmax | stokurew | gueis8850 | fr3iH3it | pearpear | wlxgjf | kambala11 |

LEAKED LISTS

Complete left lists from public leaks

| ID | Name | Last Update | Num of Hashes | Progress | Left Hashes | Found |
|------|----------------------------------|-----------------------|---------------|----------------------|---------------------|---------------------|
| 6505 | H4v3 1 b33n pwn3d (SHA1) | 02.10.2017 - 02:03:24 | 320'294'464 | 319'837'535 (99.86%) | Get | Get |
| 5638 | P4y4sUGym (MD5) | 02.10.2017 - 02:04:19 | 241'266 | 221'152 (91.66%) | Get | Get |
| 4920 | L1nk3d1n (SHA1) | 02.10.2017 - 03:24:58 | 61'829'262 | 60'147'825 (97.28%) | Get | Get |
| 3282 | 4mzr3v13w7r4d3r.c0m (MYSQL5) | 02.10.2017 - 03:25:32 | 41'823 | 39'166 (93.65%) | Get | Get |
| 3186 | X5pl17 (SHA1) | 02.10.2017 - 03:32:38 | 2'227'254 | 2'162'101 (97.07%) | Get | Get |
| 2499 | Hashkiller 32-hex left total | 02.10.2017 - 11:48:14 | 9'976'651 | 1'723'709 (17.28%) | Get | Get |
| 2498 | Hashkiller 40-hex left total | 02.10.2017 - 13:22:34 | 1'739'204 | 350'788 (20.17%) | Get | Get |
| 1619 | 4m4t3urc0mmuni7y.c0m | 02.10.2017 - 13:33:26 | 197'302 | 57'407 (29.1%) | Get | Get |
| 1535 | b73r.c0m (MD5) | 02.10.2017 - 13:34:43 | 63'070 | 32'543 (51.6%) | Get | Get |
| 1427 | 4v17r0n.fr | 02.10.2017 - 13:34:43 | 2'405 | 2'334 (97.05%) | Get | Get |
| 1366 | v0d4f0n3 (MD5(\$pass."s+_a*")) | 02.10.2017 - 13:34:44 | 322 | 307 (95.34%) | Get | Get |
| 1214 | 114_1167_5_27 (MD5) | 02.10.2017 - 13:34:44 | 176 | 80 (45.57%) | Get | Get |

Password Hashing Functions

Problem: hash functions are very fast to evaluate → facilitate fast password cracking

Solution: slow down guessing process (password “stretching”)

Benefit: cracking becomes very inefficient (e.g., 10-100ms per check)

Drawback: increased cost for the server if it must handle many users

Make heavy use of available resources

Computation should be fast enough to validate honest users, but render password guessing infeasible

Adaptable: flexible cost (time/memory complexity) parameters

Bcrypt [Provos and Mazières, 1999]

Cost-parameterized, modified version of the Blowfish encryption algorithm

Tunable cost parameter (exponential number of loop iterations)

Alternatives: Scrypt (memory-hard), PBKDF2 (PKCS standard)

Online Guessing

Similar strategy to offline guessing, but rate-limited

Connect, try a few passwords, get disconnected, repeat...

Prerequisite: know a valid user name

Many failed attempts can lead to a system reaction

Introduce delay before accepting future attempts (exponential backoff)

Shut off completely (e.g., ATM capturing/disabling a card after 3 tries)

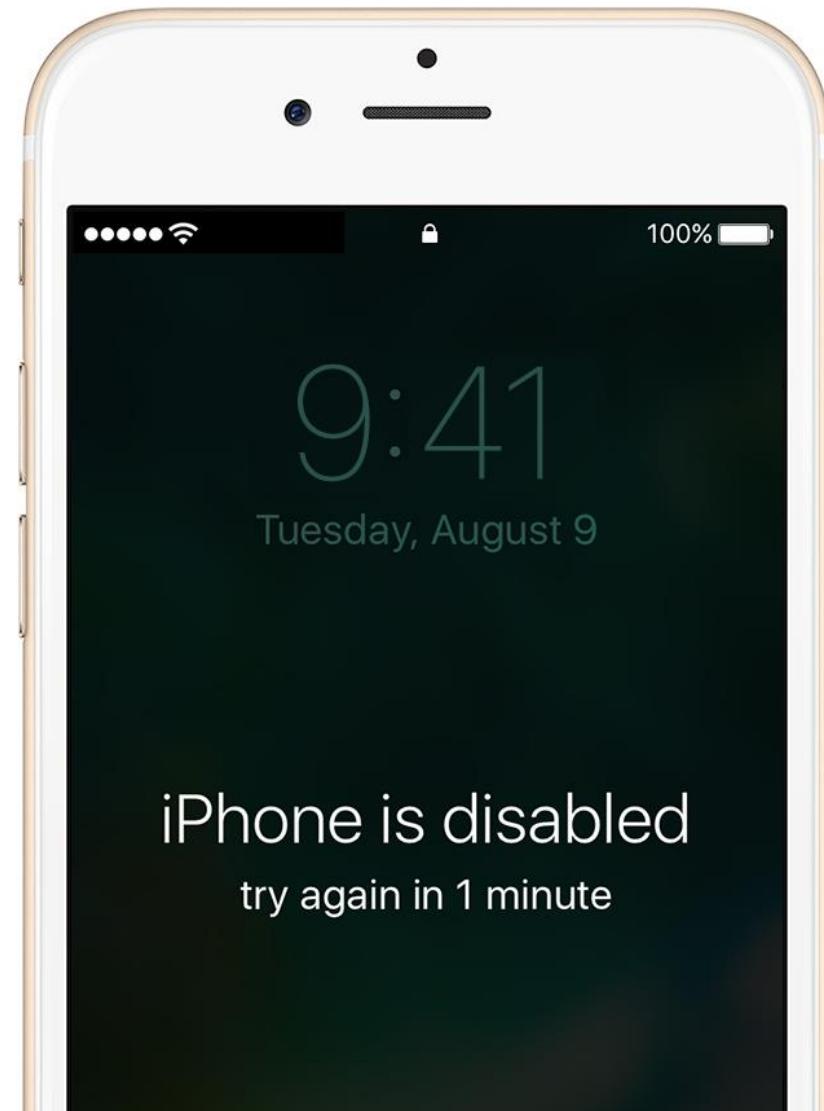
Ask user to solve a CAPTCHA

Very common against publicly accessible SSH, VPN, RDP, and other servers

Main reason people move sshd to a non-default port

Fail2Ban: block IP address after many failed attempts → may allow an attacker to lock you out of the server (!)

Better: disable password auth and use a key pair → cumbersome if having to log in from many/others' computers



LOGIN: mitch
PASSWORD: FooBar!-7
SUCCESSFUL LOGIN

(a)

LOGIN: carol
INVALID LOGIN NAME
LOGIN:

(b)

LOGIN: carol
PASSWORD: Idunno
INVALID LOGIN
LOGIN:

(c)

(a) A successful login

(b) Login rejected after name is entered

(c) Login rejected after name and password are typed → less information makes guessing harder

Try the Default First

The screenshot shows a web browser window for 'Default Router Passwords' at www.routerpasswords.com. The page title is 'RouterPasswords.com'. The main content area displays a table of default router passwords for Cisco devices. The table has columns for Manufacturer, Model, Protocol, Username, and Password. The 'Manufacturer' column is bolded.

| Manufacturer | Model | Protocol | Username | Password |
|--------------|-----------------------------------|-----------------------|-------------|-----------|
| CISCO | CACHE ENGINE | CONSOLE | admin | diamond |
| CISCO | CONFIGMAKER | | cmaker | cmaker |
| CISCO | CNR Rev. ALL | CNR GUI | admin | changeme |
| CISCO | NETRANGER/SECURE IDS | MULTI | netrangr | attack |
| CISCO | BBSM Rev. 5.0 AND 5.1 | TELNET OR NAMED PIPES | bbsd-client | changeme2 |
| CISCO | BBSD MSDE CLIENT Rev. 5.0 AND 5.1 | TELNET OR NAMED PIPES | bbsd-client | NULL |

Eavesdropping and Replay

Physical world

Watch user type password (shoulder surfing)

Cameras (ATMs skimmers)

Lift fingerprints (iPhone)

Post-it notes

Network makes things easier

Sniffing (LAN, WiFi, ...)

Man-in-the-Middle attacks

Defenses

Encryption

One-time password schemes

Kerberos

Long-lived vs. session keys

Use long-lived key for authentication and negotiating session keys

Use “fresh,” ephemeral session keys (prevent replay, cryptanalysis, old compromised keys) for encrypted communication, MACs, ...

Kerberos: most widely used (non-web) single sign-on system

Originally developed at MIT, now used in Unix, Windows, ...

Authenticate users to services: using their password as the initial key, without having to retype it for every interaction

A Key Distribution Center (KDC) acts as a trusted third party for key distribution

Online authentication: Variant of Needham-Schroeder protocol

Assumes a non-trusted network: prevents eavesdropping

Assumes that the Kerberos server and user workstations are secure...

Use cases: workstation login, remote share access, printers, ...

Password Capture

Hardware bugs/keyloggers

Software keyloggers/malware

Cameras

Phishing

Social engineering

Welcome to



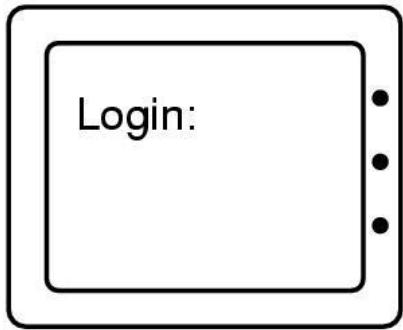
Microsoft®
Windows®
Professional

Copyright © 1999-2001
Microsoft Corporation

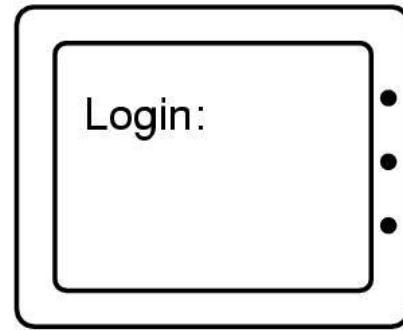


Press Ctrl-Alt-Delete to begin.

Requiring this key combination at startup helps keep
computer secure. For more information, click Help.



(a)



(b)

- (a) Correct login screen
(b) Phony login screen

Something You Have: Authentication Tokens

One-time passcode tokens

Time-based

Counter-based



Other authentication tokens: store certificates, encryption keys, challenge-response, ...

Smartcards (contact or contactless)

Identification, authentication, data storage, limited processing

Magnetic stripe cards, EMV (chip-n-pin credit cards), SIM cards, RFID tags, ...

USB/NFC tokens, mobile phones, watches, ...

Can be used as authentication devices

Multi-factor Authentication

Present several separate credentials of different types

Most common: *two-factor authentication (2FA)*

Example: Password + hardware token, mobile phone, ...

Example: ATM card + PIN

Motivation: a lost/guessed password is not enough anymore for attackers → not always true

Man-in-the-Middle: set up fake banking website, relay password to real website, let the user deal with the second factor...

Man-in-the-Browser: hijack/manipulate an established session after authentication has completed (banking Trojans)

Dual infection: compromise both PC and mobile device

Implementation-dependent usability issues

Token may be lost, in-flight WiFi but cannot receive SMS, ...

Fallback: backup one-time-use passcodes (where to keep them?)

SMS Is Not a Secure 2nd Factor

(but still better than no 2nd factor)

Social engineering

Call victim's mobile operator and hijack the phone number

SIM swap, message/call forwarding, ...

Message interception

Rogue cell towers: IMSI catchers, StingRays,...

Some phones even display text messages on the lock screen (!)

SS7 attacks

The protocol used for inter-provider signaling is severely outdated and vulnerable

Allows attackers to spoof change requests to users' phone numbers and intercept calls or text messages



2-Step Verification

A text message with your code has been sent to:
*** * * * 67

Enter code

Verify

Remember this computer for 30 days.





sign in



become a supporter

subscribe



search

jobs

US edition

the guardian



US politics

world

opinion

sports

soccer

tech

arts

lifestyle

fashion

business

travel

environment

all sections

home

Scams

'Sim swap' gives fraudsters access-all-areas via your mobile phone

There's a new, little-known scam designed to empty your bank account, as one Vodafone customer found to her cost



1908 15

Anna Tims

Saturday 26 September
2015 02.00 EDT



Most popular in US



Las Vegas shooting:
death toll rises to 58 as
police name suspect –
latest updates



Confusion follows
reports of Tom Petty
death after heart attack



Las Vegas gunman may
have used special device
to fire faster, expert says

Better Alternative: Authenticator App

Six/eight digit code provided after successful password validation

Time-based one-time password (TOTP)

Code computed from a shared secret key and the current time (using HMAC)

The key is negotiated during registration

Requires “rough” synchronization between client and server

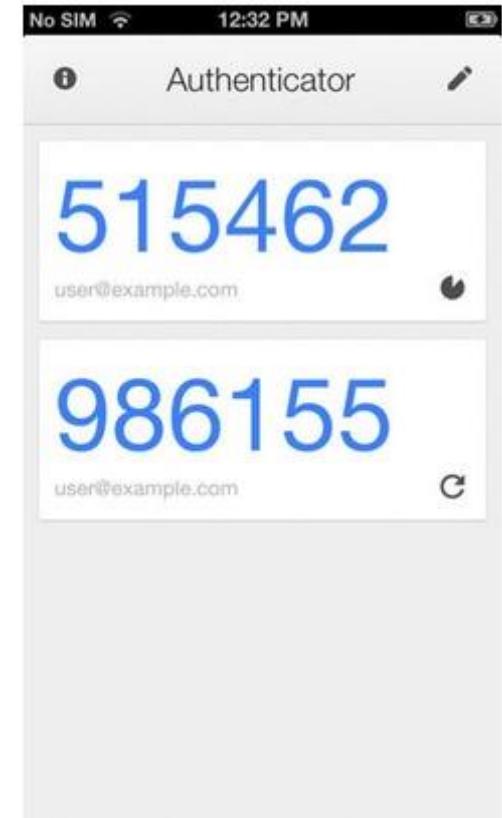
Code constantly changes in 30-second intervals

Phishing is still possible!

The attacker needs to proxy the captured credentials in real time (rather than collecting them for later use)

Session hijacking and Man-in-the-Middle are still possible!

After the user has successfully logged in



Even Better Alternative: U2F Tokens

Universal Second Factor (U2F)

FIDO (Fast IDentity Online) alliance: Google, Yubico, ...

Supported by many popular online services

Supported by Chrome and Opera
(and soon Mozilla and Microsoft)



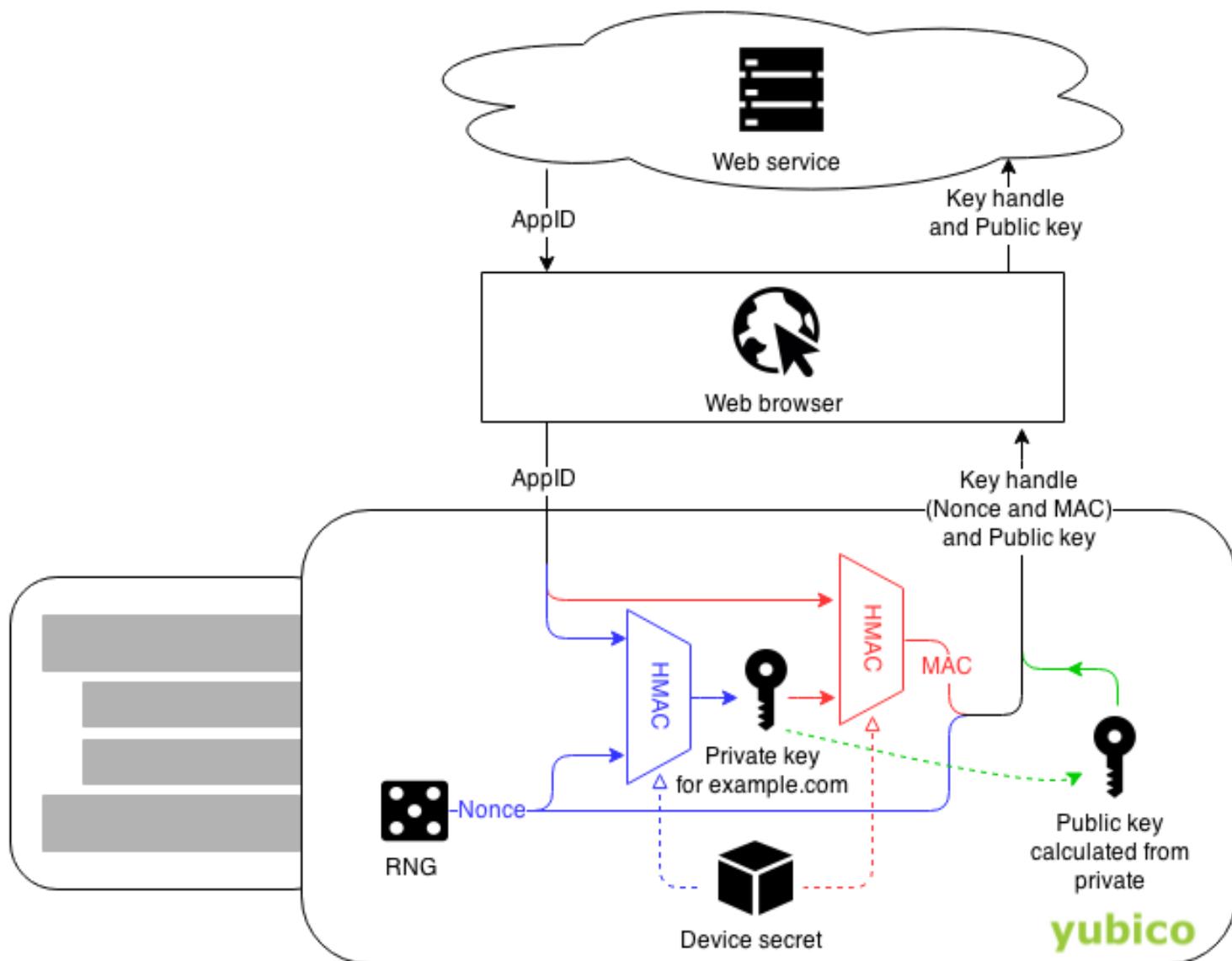
A different key pair is generated for each origin during registration

Origin = <protocol, hostname, port>

Private key stored on device

Public key sent to server





U2F tokens

Benefits

Easy: just tap the button (no typing)

Works out of the box (no drivers to install)

USB, NFC, Bluetooth communication

No shared secret between client and server

Origin checking → effective against phishing!



Drawbacks

Can be lost → a fallback is needed (e.g., Authenticator App)

Still a bit cumbersome: have to pull keychain out of pocket and plug token in (or have an always pugged-in token per device)

Cost (\$7 – \$60)

Man-in-the-Browser is still possible!

Single Sign-on/Social Login

Pros

Convenience: fewer passwords to remember

Rich experience through social features

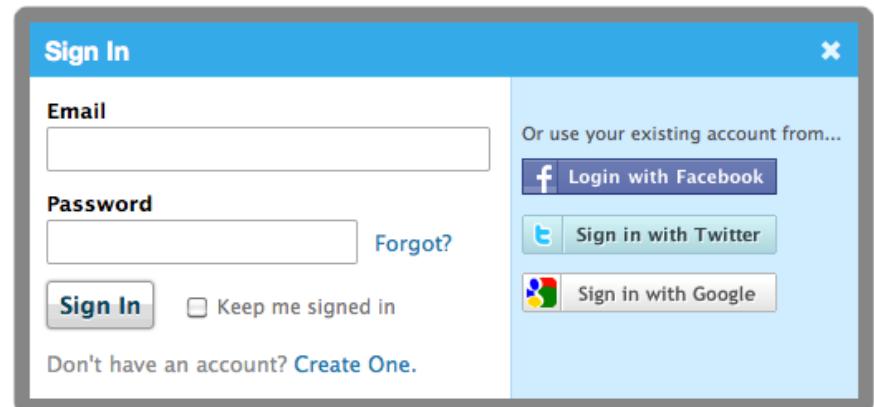
Easier development: outsource user registration/management

Cons

Same credentials for multiple sites: single point of failure

Access to user's profile

User tracking



Biometrics

Fingerprint reader

Face recognition

Depth sensing, “liveness” detection (pulse, thermal), etc. to foil simple picture attack

Retina/iris scanner

Voice recognition

...

Continuous authentication

Keystroke timing, usage patterns, ...

Crypto-based Authentication

Rely on a cryptographic key to prove a user's identity

User performs a requested cryptographic operation on a value (challenge) that the verifier supplies

- Usually based on knowledge of a key (secret key or private key)

- Can use symmetric (e.g., Kerberos) or public key schemes

How can we trust a key? Why is it authentic?

- Need to establish a level of trust

Different approaches: **TOFU, PKI, web of trust**

- Emerging approach: blockchain/ledger-based PKI

Trust on First Use (aka Key Continuity)

Use case: SSH

Performs *mutual authentication*

Server *always* authenticates the client
password, key pair, ...

Client almost always authenticates the server – *except the first time!*

First connection: server presents its public key

No other option for the user but to accept it: MitM opportunity

Subsequent connections: client remembers server's key, and triggers an alert on key mismatch

Pragmatic solution, but shifts the burden to users

Users must determine the validity of the presented key

Assuming a key change is valid without verifying the new key offers no protection against MitM (unfortunately, that's what most users do)

@@oo

@ WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED! @

@oo

IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!

Someone could be eavesdropping on you right now (man-in-the-middle attack)!

It is also possible that the RSA host key has just been changed.

The fingerprint for the RSA key sent by the remote host is

df:c8:52:aa:cd:e3:da:8c:ec:50:46:db:4d:21:d9:c7.

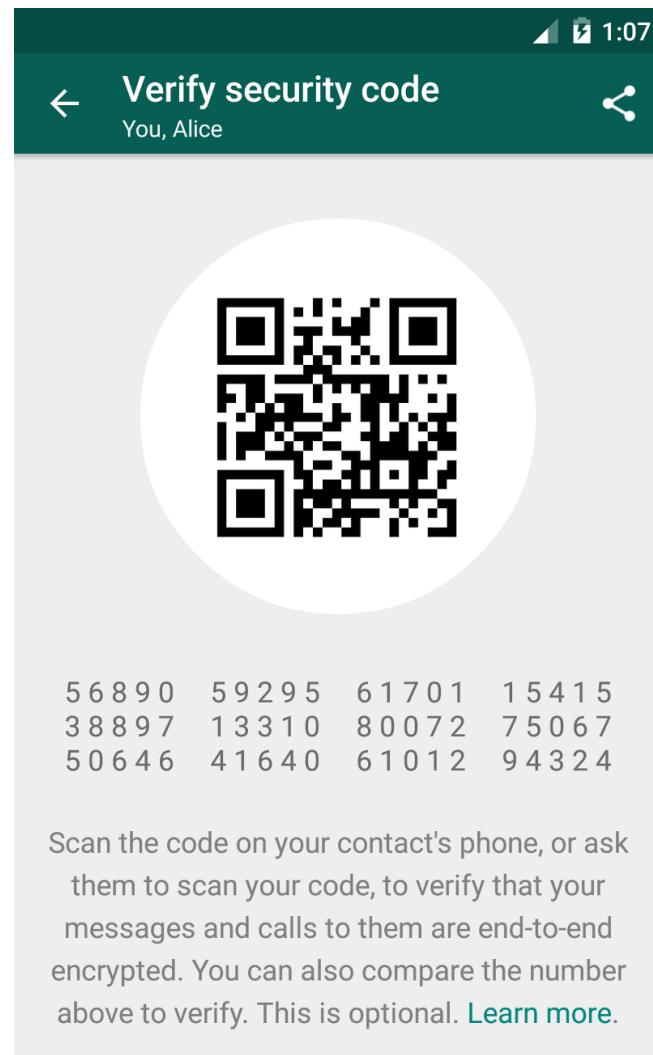
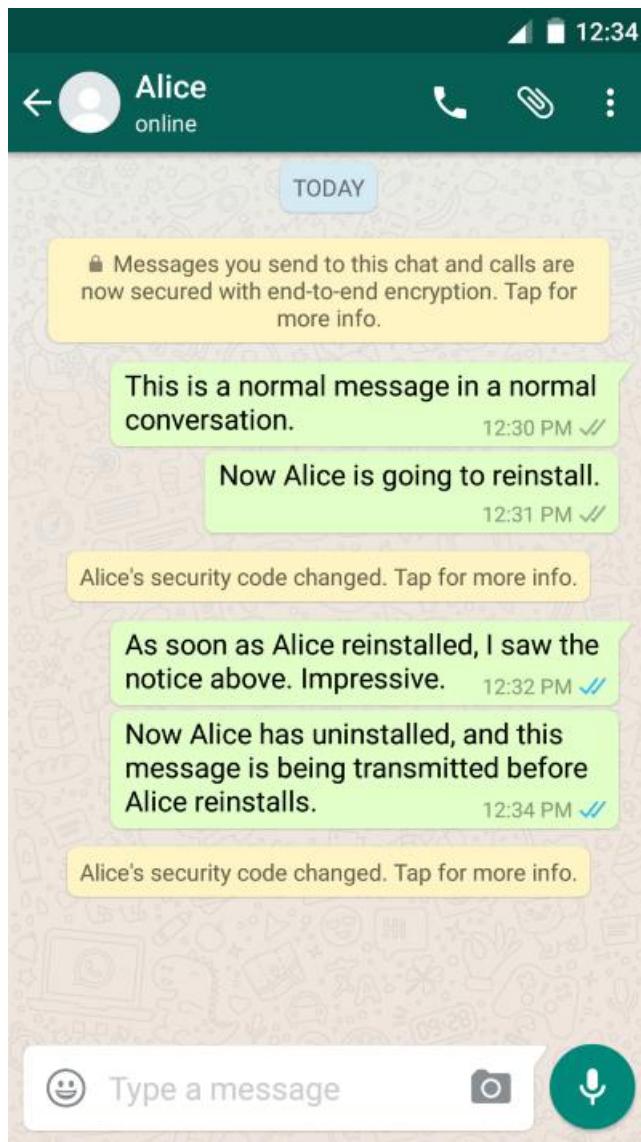
Please contact your system administrator.

Add correct host key in /root/.ssh/known_hosts to get rid of this message.

Offending key in /root/.ssh/known_hosts:1

RSA host key for 192.168.2.5 has changed and you have requested strict checking.

Host key verification failed.



Certificates

How can we distribute “trusted” public keys?

Public directory → risk of forgery and tampering

More practical solution: “certified” public keys

A certificate is a digitally signed message containing an identity and a public key

Makes an association between a user/entity and a private key

Valid until a certain period

Why trust a certificate?

Because it is signed by an “authority”

Third party’s signature prevents tampering



Public Key Infrastructures (PKI)

Facilitate the authentication and distribution of public keys based on identities

Set of roles, policies, and procedures to create, manage, distribute, use, store, and revoke certificates

An issuer signs certificates for subjects

Trust anchor

Methods of certification

Certificate authorities (hierarchical structure – root of trust)

Web of trust (decentralized, peer-to-peer structure)

Certificate Authorities

Trusted third-parties responsible for certifying public keys

Most CAs are tree-structured

Single point of failure: CAs can be compromised!

Why should we trust an authority?

How do you know the public key of the Certificate Authority (CA)?

CA's public key (trust anchor) must somehow be provided out of band

Trust has to start somewhere

Operating systems and browsers are pre-configured with ~200 trusted root certificates

A public key for any website in the world will be accepted without warning if certified by any of these CAs (more in the TLS lecture)



Secure | https://www.wired.com/2011/09/diginotar-bankruptcy/



WIRED

DigiNotar Files for Bankruptcy in Wake of Devastating Hack

SUBSCRIBE



KIM ZETTER SECURITY 09.20.11 03:05 PM

SHARE



DIGINOTAR FILES FOR BANKRUPTCY IN WAKE OF DEVASTATING HACK



Go to ...

A Dutch certificate authority that suffered a major hack attack this summer has been unable to recover from the blow and filed for bankruptcy this week.

MOST POPULAR

TRANSPORTATION

General Motors Announces an All-Electric Future
ALEX DAVIES



SECURITY
This "Ghost Gun" Machine Now Makes Untraceable Metal Handguns
ANDY GREENBERG



SECURITY
How the Las Vegas Shooter Could Have Gotten an Automatic Rifle
ANDY GREENBERG



MORE STORIES

Web of Trust

Entirely decentralized authentication

- No single point of failure

- No need to buy certs from CAs

- Used in PGP

Users sign other users' keys

- Only if they deem them trustworthy

- Certificate signings can form an arbitrarily complex graph

- Users can verify path to as many trust anchors as they wish

Drawbacks

- Hard to use, requires in-person verification – key signing parties!

- Hard to know what trust level to assign transitively

WoT Alternative: Online Social “Tracking”

The screenshot shows a web browser window displaying the Keybase.io profile page for user [mikepo](https://keybase.io/mikepo). The URL in the address bar is `https://keybase.io/mikepo`. The page features a dark header with the Keybase logo, a search bar, and links for 'Join' and 'Login'. Below the header is a sidebar for the user **Michalis Polychronakis**, which includes a circular profile picture and the name **Michalis Polychronakis**. To the right of the sidebar, the main content area displays the user's public key fingerprint `8EBD 8F30 8899 8AFF` and social links for **polychronakis** on Twitter and GitHub. A green callout box informs the user that they have an invitation available. At the bottom left, there are buttons for 'Encrypt' and 'Verify'. On the right side, there are two sections: 'Tracking (6)' and 'Trackers (6)', each listing six users with their profile pictures and names.

keybase.io/mikepo

8EBD 8F30 8899 8AFF

polychronakis tweet

polychronakis gist

mikepo has an invitation available
If you know mikepo, you can ask them for an invitation to Keybase.

Encrypt Verify

mikepo from the command line

```
# first
keybase join # if you're new, or
keybase login # if you're not.

# then
keybase push # if you already have a public key, or
keybase gen # if this is all new to you
```

Tracking (6)

| Profile Picture | Name |
|-----------------|-----------------|
| | hargikas |
| | mstamat |
| | gianluca_string |
| | hargikas |
| | kontaxis |
| | mstamat |

Trackers (6)

Keybase.io

In essence, a directory associating public keys with names

Identity established through *public signatures*

Identity proofs: “I am Joe on Keybase and MrJoe on Twitter”

Follower statements: “I am Joe on Keybase and I just looked at Chris's identity”

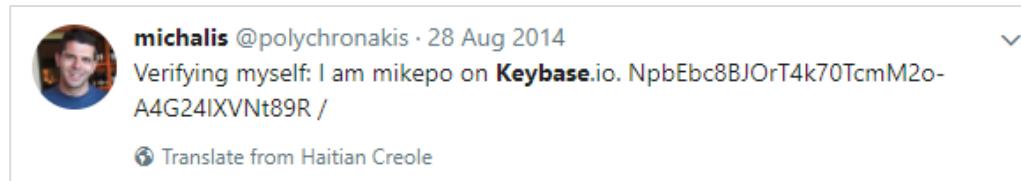
Key ownership: “I am Joe on Keybase and here's my public key”

Revocations: “I take back what I said earlier”

Keybase identity = sum of other public identities

Twitter, Facebook, Github, Reddit, domain ownership, ...

Example:



michalis @polychronakis · 28 Aug 2014
Verifying myself: I am mikepo on Keybase.io. NpbEbc8BJOrT4k70TcmM2o-A4G24IXVNt89R /
.Translate from Haitian Creole

An attacker has to compromise all connected identities

The more connected identities, the harder to impersonate a user

Best Practices

Pick long passwords (passphrases)

Never reuse the same password on different services

Never share passwords

Use SSH keys instead of passwords

Use two-factor authentication when available

Use Authenticator App or U2F instead of SMS

Disassociate phone number from account after initial setup

Use a password manager

Not only for passwords! Also for “security” questions...