CSE508 Network Security

3/2/2016

Authentication

Michalis Polychronakis

Stony Brook University

Authentication

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

What is identity?

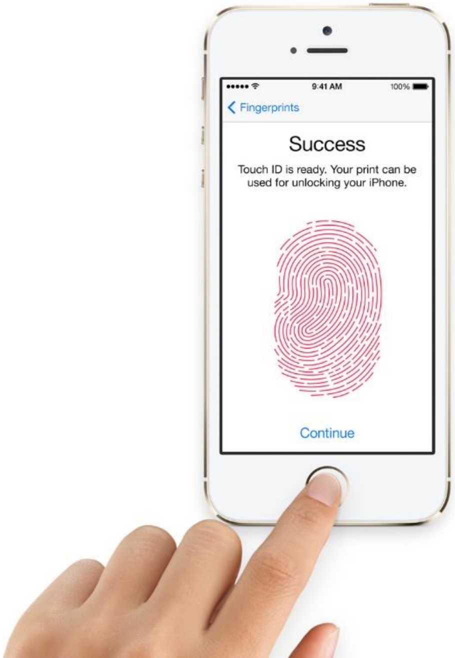
- Which characteristics uniquely identify an entity?

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



Success

Touch ID is ready. Your print can be used for unlocking your iPhone.

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

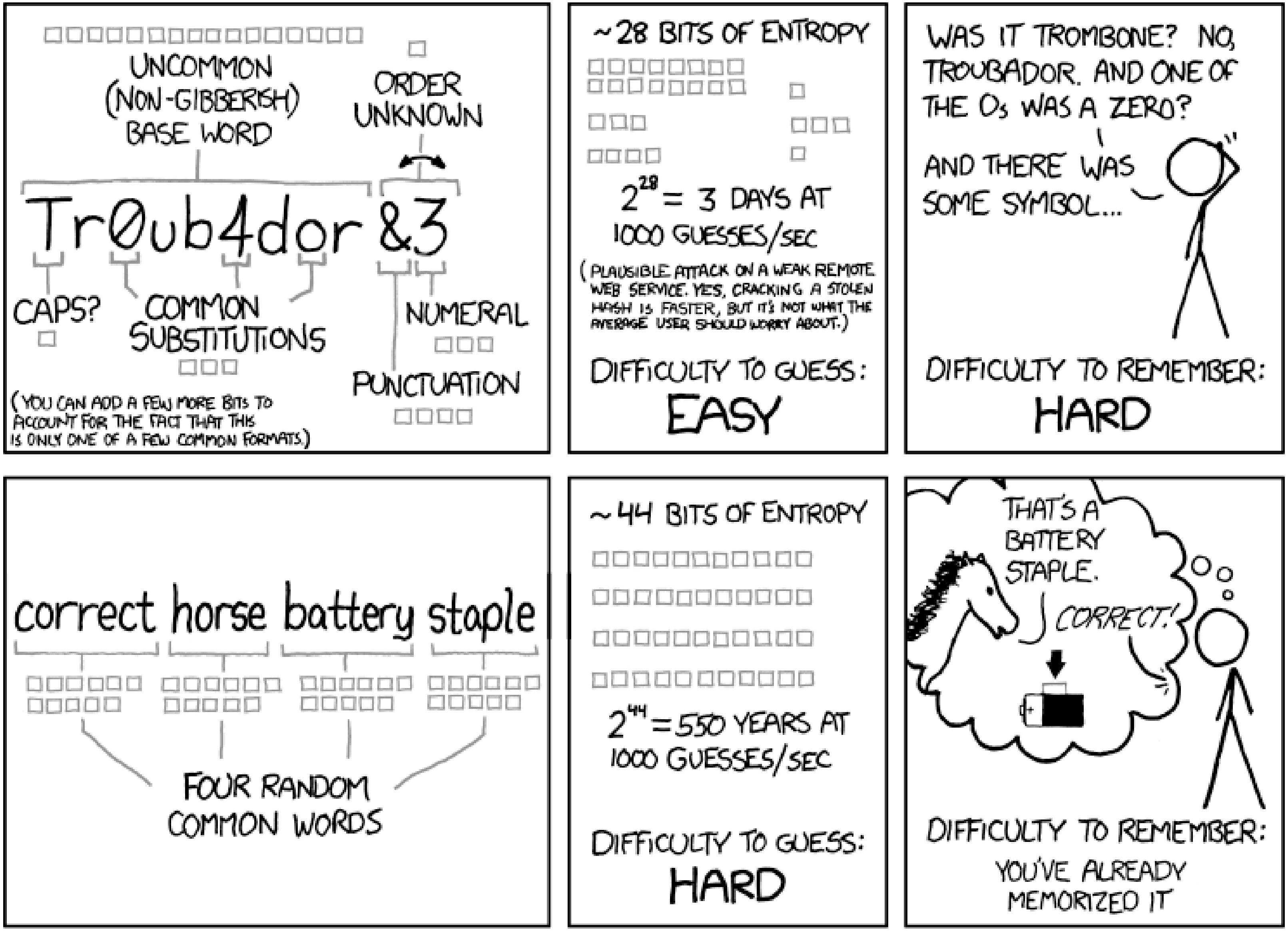
- Easy to remember ➔ easy to crack
- Hard to crack ➔ hard to remember
- Bad ideas: DOB, SSN, zip code, favorite team name, …

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, …


UNCOMMON (NON-GIBBERISH) BASE WORD
ORDER UNKNOWN
Tr0ub4dor &3
CAPS?
COMMON SUBSTITUTIONS
NUMERAL
PUNCTUATION
(YOU CAN ADD A FEW MORE BITS TO ACCOUNT FOR THE FACT THAT THIS IS ONLY ONE OF A FEW COMMON FORMATS.)
~28 BITS OF ENTROPY
2^28 = 3 DAYS AT 1000 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 Os WAS A ZERO?
AND THERE WAS SOME SYMBOL...
DIFFICULTY TO REMEMBER: HARD
correct horse battery staple
FOUR RANDOM COMMON WORDS
~44 BITS OF ENTROPY
2^44 = 550 YEARS AT 1000 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.

© XKCD - https://xkcd.com/936/

Attacking Passwords

Offline cracking

Online guessing

Brute force attacks

Eavesdropping

Capturing

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

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 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
- List of previously leaked real user passwords

Variations and common patterns

- Prepend/append symbols/numbers/dates, weird capitalization, l33tspeak, visually similar characters, intended misspellings, …

Target-specific information

- DOB, family names, favorite team, pets, hobbies, anniversaries, language, slang, …
- Many ease to acquire from social networking services
- Particularly effective against "security questions"

Advanced techniques

- Probabilistic context-free grammars, Markov models, …

Combination of all the above

25 Most-used (Worse) Passwords

password

123456

12345678

1234

qwerty

12345

dragon

pussy

baseball

football

letmein

monkey

696969

abc123

mustang

michael

shadow

master

jennifer

111111

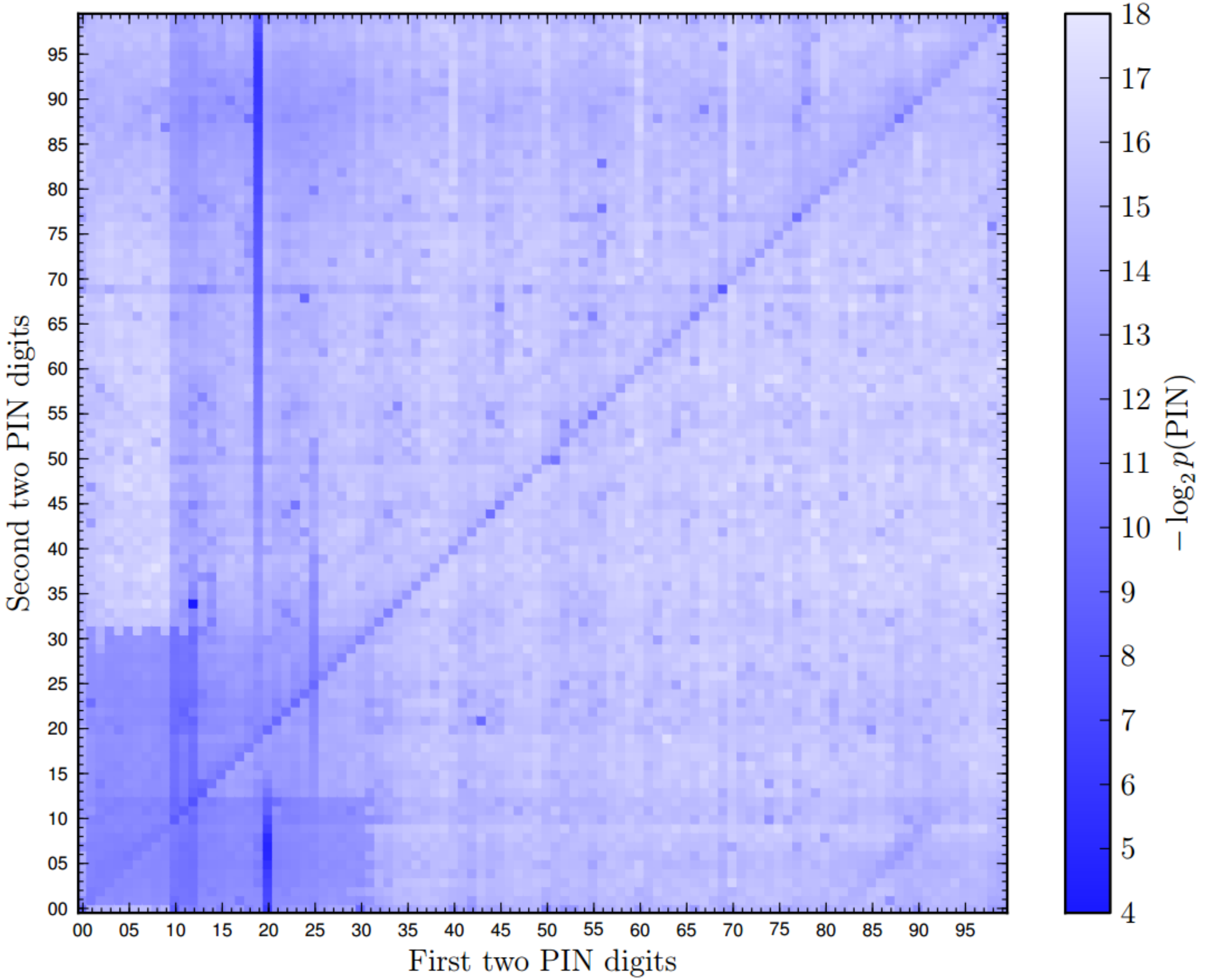
2000

jordan

superman

harley

1234567



Distribution of 4-digit sequences within RockYou passwords

A birthday present every eleven wallets? The security of customer-chosen banking PINs. Joseph Bonneau, Sören Preibusch and Ross Anderson – FC '12

Wordlists

```
ce#ebc.dk
goddess5
20071002
271075711
zs3cu7za
scoopn
frygas1411
SL123456sl
12345687ee123
xuexi2010
daigoro
12345614
DICK4080
567891234
tilg80
6z08c861
:zark:
ravishsneha
150571611369
661189
passme
trolovinasveta
abdulkhaleque
007816
xLDSX
Florida2011
037037
WestC0untry
hitsugaiya
955998126
3n3rmax
4637324
bugger825
marmaris
jinjin111
170383gp
3484427
fl33321
zwqrfg
67070857
432106969
6856
704870704870
pv041886
20060814
512881535
milanimilani
472619
dbyxw888
85717221
cc841215
ariana19321
bbbnnn
ang34hehiu
wj112358
Brenda85
786525pb
shi461988
pingu
yeybozip
71477nak
stokurew
gea8mw4yz
kukumbike
260888
jordi10
lexusis
kj011a039
c84bwlrb
priyanka05
loveneverdies
u8Aqebj576
FGYfgy77
659397
327296
74748585
19720919
050769585
nicopa
2232566
bearss
n0tpublic
isitreal00
ashraf19760
48144
22471015
antyzhou115
0167005246
ec13kag
226226226226
6767537/33
mimilebrock
gueis8850
fujinshan
counter
N8mr0n
520057
adc123
bmaster
qbjh04zg
ueldaa79
EMANUELLI
yanjing
assynt
62157173
0704224950753
6903293
axaaxa
hilall
30091983
2510618981
soukuokpan
tosecondlife
p4os8m6q
015614117
acw71790
lsyljm2
2xgialdl
gaybar9
88203009
MKltyh87
quiggle
2063775206
fr3iH3it
masich
pengaiwei
coalesce
56402768
thesis
aabbcc894
marion&maxime
614850
ydz220105
584521584521
txudecp
84410545
pietro.chiara
jman1514
heryarma
39joinmam
timelapse
mwinkar
251422
willrock
YHrtfgDK
xys96exq
mercadotecnia
8s5sBEx7
0125040344
margitka
omaopa
dfTi6nh
1314520521
pixma760
pearpear
gothpunksk8er
rftaeo48
8d7R0K
5172032
aics07
34mariah
dongqinwei
samarica
cap10l4
0167387943
AE86Trueno
19700913
mcsuap
bu56mpbu
danbee
passw<>
money521
conan83
nxfjpl
rateg143
kojyihen
058336257
sarah4444
7363437
freindship
JytmvWO848
sb inbau
30907891
0515043111
1973@ati
wlxgjf
20081010
leelou44
8UfjeGbO
200358808
dellede
liang123.
captainettekt
kwiki-mart
mdovydas
tigmys2001
denial
678ad5251
woaiwuai
1591591591212
hNbDGN
cardcap
13985039393
001104
desare11
412724198
nibh1kab
asferg
hqb555
xgames7
muckerlee
choqui67
12130911
lierwei120
skytdvn
milena1995
kambala11
```

Password Hashing Functions

Problem: hash functions are very fast to evaluate ➔ enable fast guessing attacks

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

- E.g., 10-100ms per check ➔ significant 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…

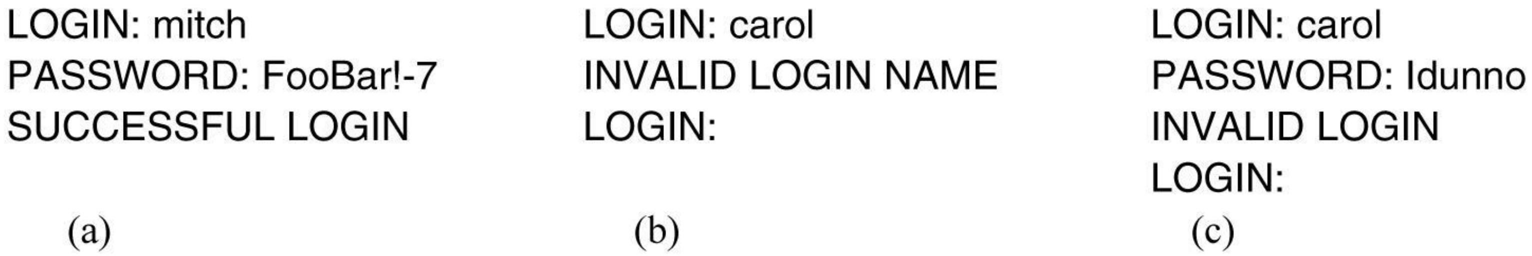
Prerequisite: know a valid user name

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:
```

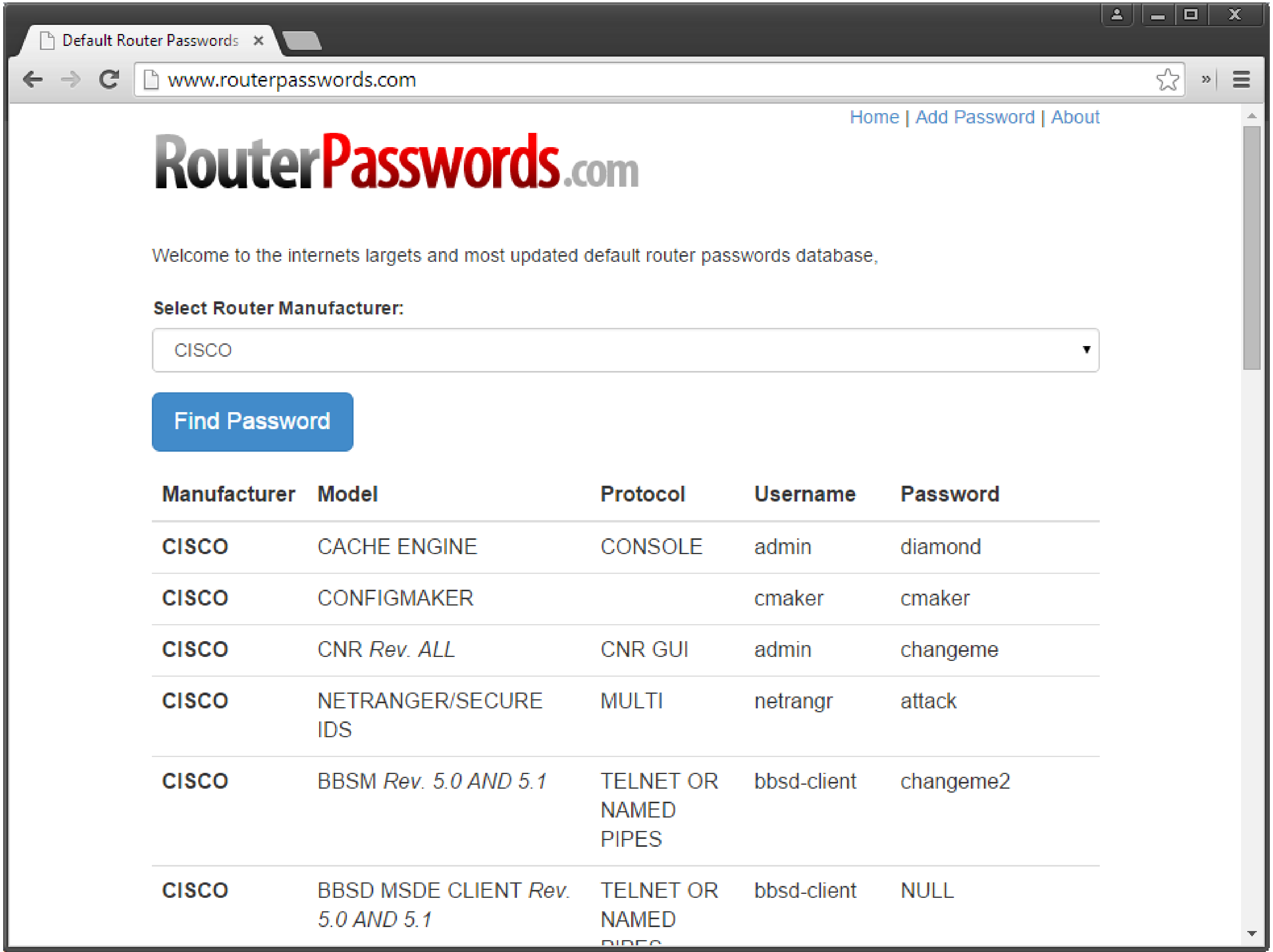
(c)

(a) A successful login

(b) Login rejected after name is entered

(c) Login rejected after name and password are typed

Try the Default First



Welcome to the internets largets and most updated default router passwords database,

Select Router Manufacturer: CISCO

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	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, …

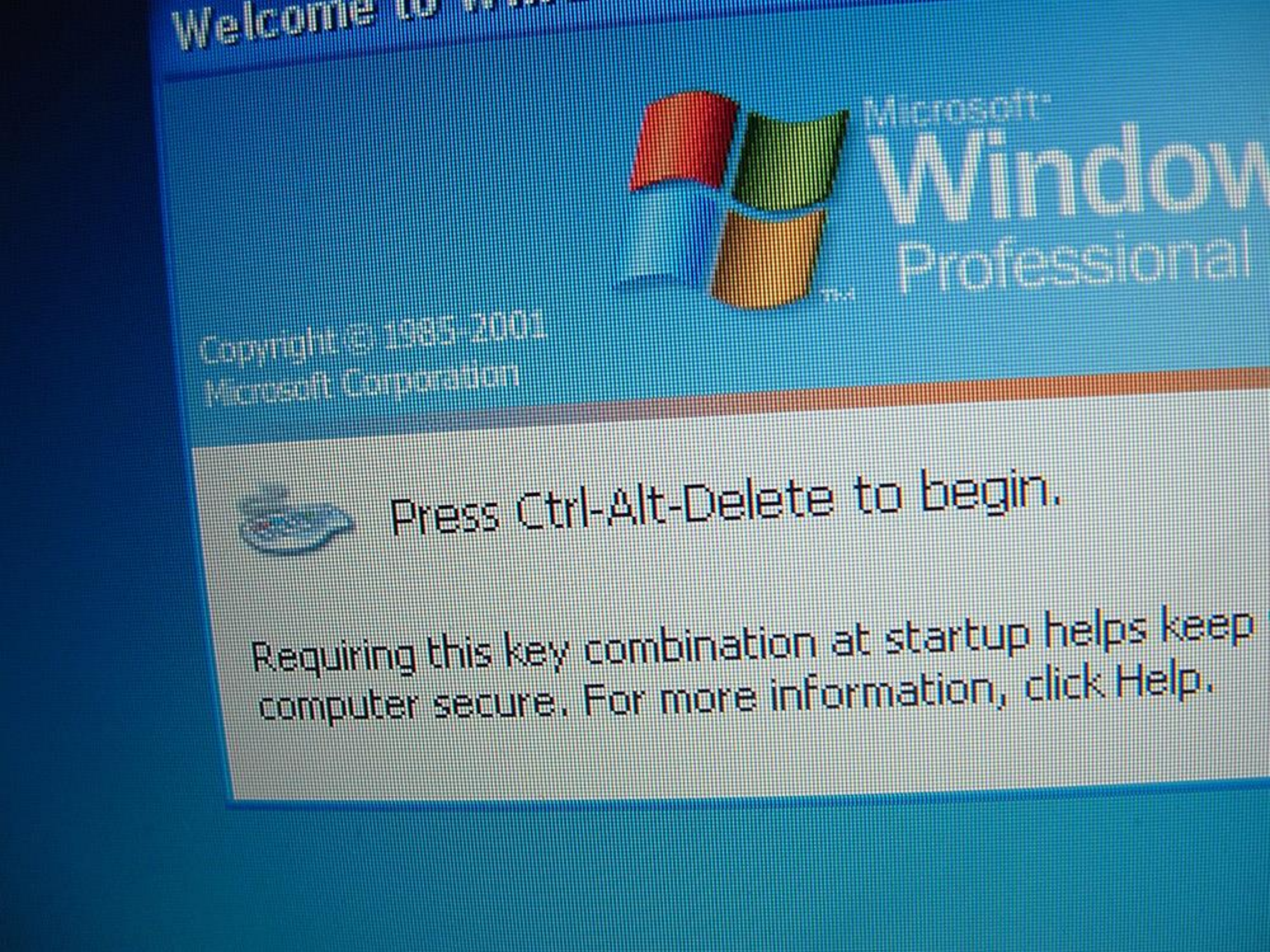
Password Capture

Hardware bugs/keyloggers

Software keyloggers/malware

Phishing

Social engineering



Copyright © 1985-2001
Microsoft Corporation

Press Ctrl-Alt-Delete to begin.

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

Login:
(a)
Login:
(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/bluetooth tokens, mobile phones, watches, …

- Can be used as authentication devices

Multi-factor Authentication

Present several separate credentials of different types

Most common: 2-factor authentication (2FA)

- Example: Password + hardware token, mobile phone, …
- Example: ATM card + pin

Motivation: a lost/guessed password is now not enough 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

- In-flight WiFi, but cannot receive SMS…
- Fallback: backup one-time-use passcodes (where to keep them?)

Biometrics

Fingerprint reader (iOS)

Face recognition (android)

Retina/Iris scanner

Voice recognition

…

Continuous authentication

Keystroke timing, usability patterns, …

Crypto-based Authentication

Some way to use a cryptographic key to prove a user's identity

Basic idea: user performs a requested cryptographic operation on a value (a 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

Use case: SSH

- Performs *mutual authentication*

Server *always* authenticates the client

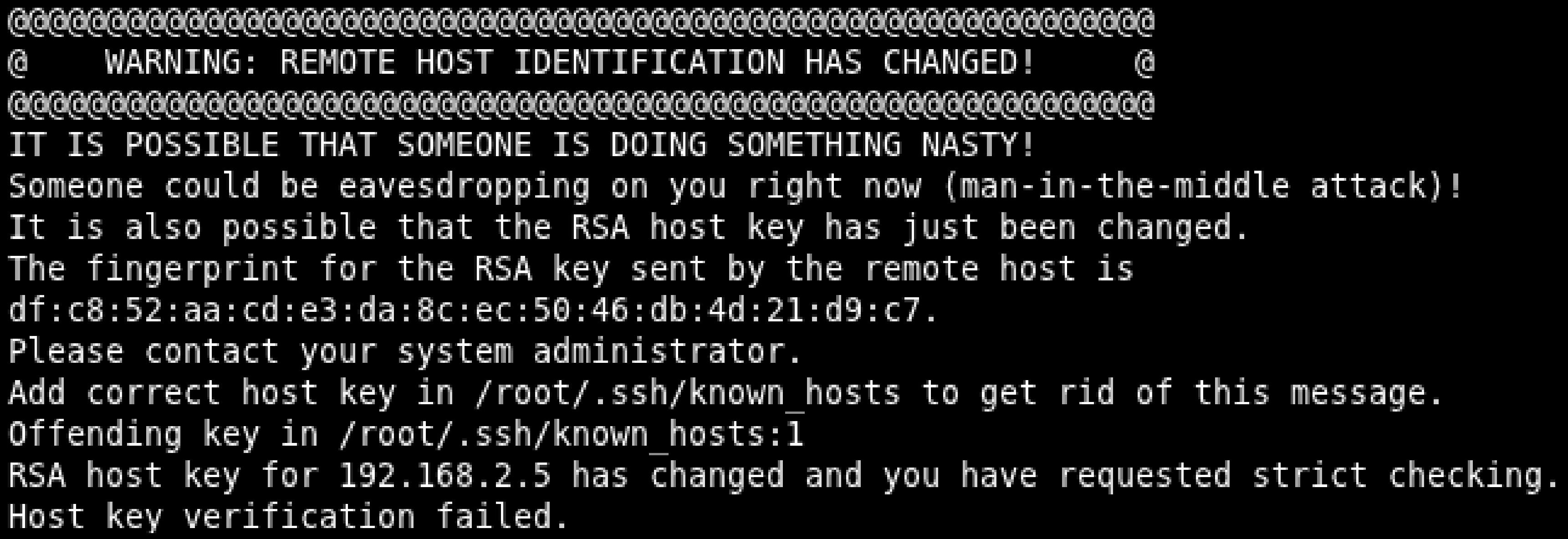
- password, key pair, …

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)



```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
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, mange, 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 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

Web of Trust

Entirely decentralized authentication

- No single point of failure
- No need to by certs from CAs
- Used in PGP

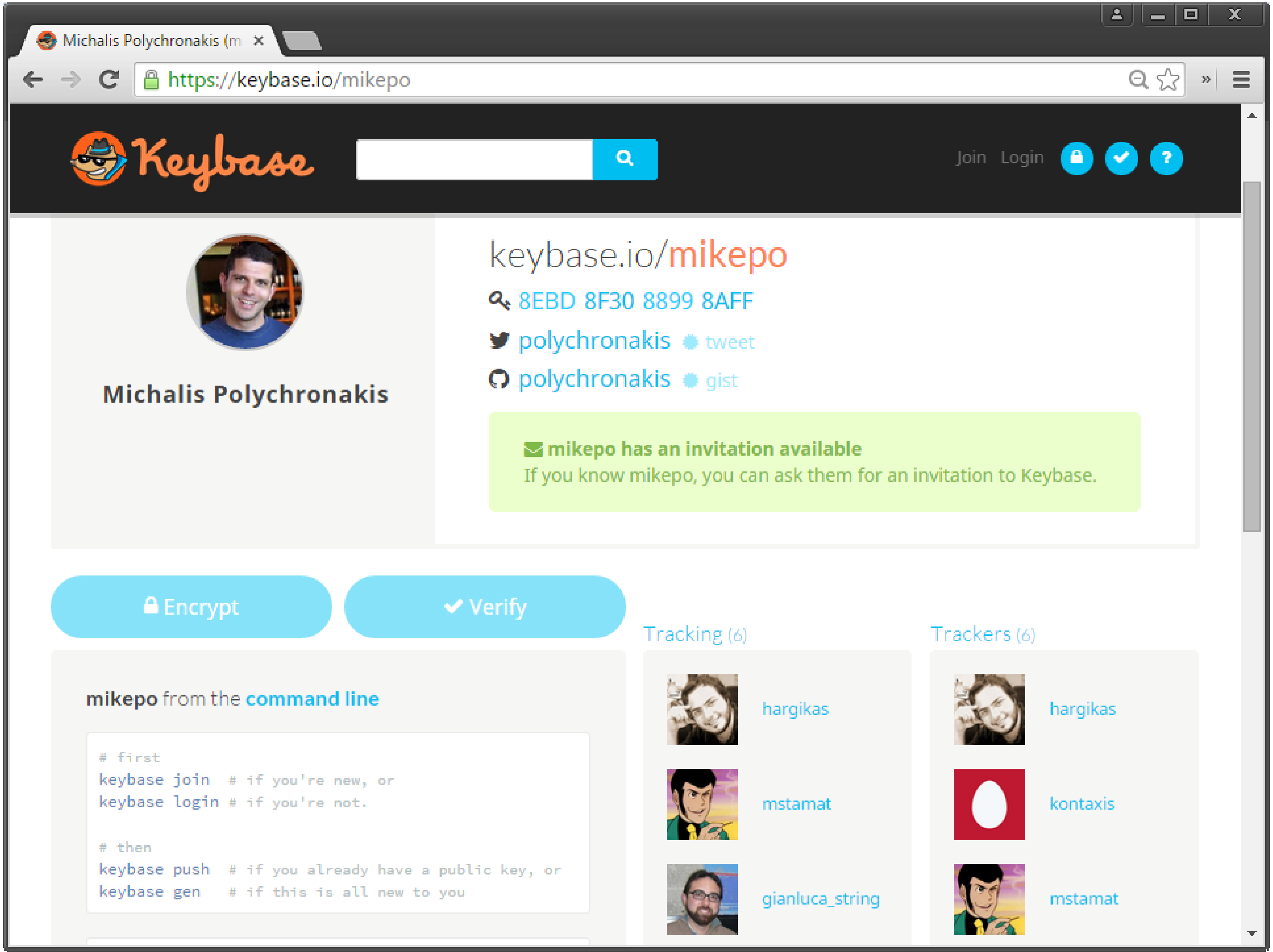
Users sign other users' keys

- 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"



Michalis Polychronakis

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.

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)

hargikas
mstamat
gianluca_string

Trackers (6)

hargikas
kontaxis
mstamat

Best Practices

User education is important!

Pick long passwords (passphrases)

Never reuse the same password on different services

Never share passwords

No post-its

Use two-factor authentication when possible

Use a password manager

- Not only for passwords! Also for “security” questions…

Use SSH keys instead of passwords