Password Cracking

Adopting a reasonable password hygiene

Didier Guillevic
didier.guillevic.net

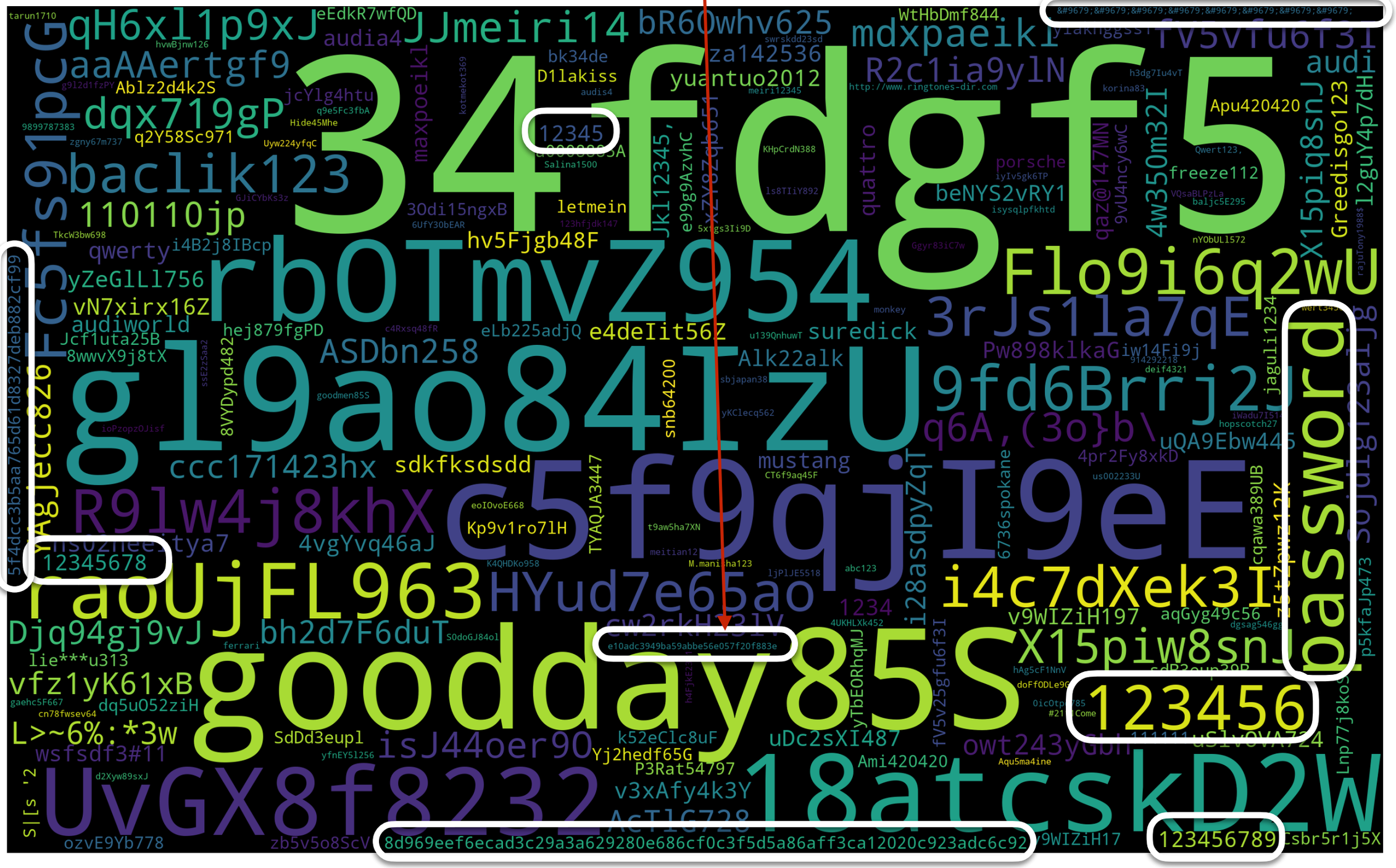
Cracked Passwords

Data from hashes.org

MD5(‘**password**’) -> 5f4dcc3b5aa765d61d8327deb882cf99

MD5(‘**123456**’) -> e10adc3949ba59abbe56e057f20f883e

unicode(●●●●●●●●)



SHA256(‘**123456**’) -> 8d969eef6ecad3c29a3a629280e686cf0c3f5d5a86aff3ca12020c923adc6c92

Summary - TL;DR

How passwords are stored

0. Plain text	"secret password”	—>	**“secret password”**
1. Hashed	sha256("secret password”)	—>	**“1ba133eccdf…8557”**
2. Salted Hash	sha256("secret password" + ”**B7aU3L**")	—>	“**a1296cc4833…6420**"
3. Key stretching	pbkdf2(“sha256”, “secret password”, ”**B7aU3L**”, **100,000**)	—>	“**0274d582e1b…82a4**”

How passwords are cracked

1. Brute force (short password, network of computers, …)
2. Use of lexicons + hand crafted rules on human behaviour when creating passwords
3. Language models (deep neural networks) (to generate candidate passwords)
4. Generative Adversarial Networks (to generate candidate passwords)

Advice

1. Use a different password for each account / web site
2. Use a passphrase instead of a password (Snowden, Better passwords, Information entropy)
3. Use a password management tool (e.g. 1Password, Bitwarden (open source), NordPass)
4. Use two factor identification (e.g. FreeOTP, Authy, Google Authenticator)

Password Storage

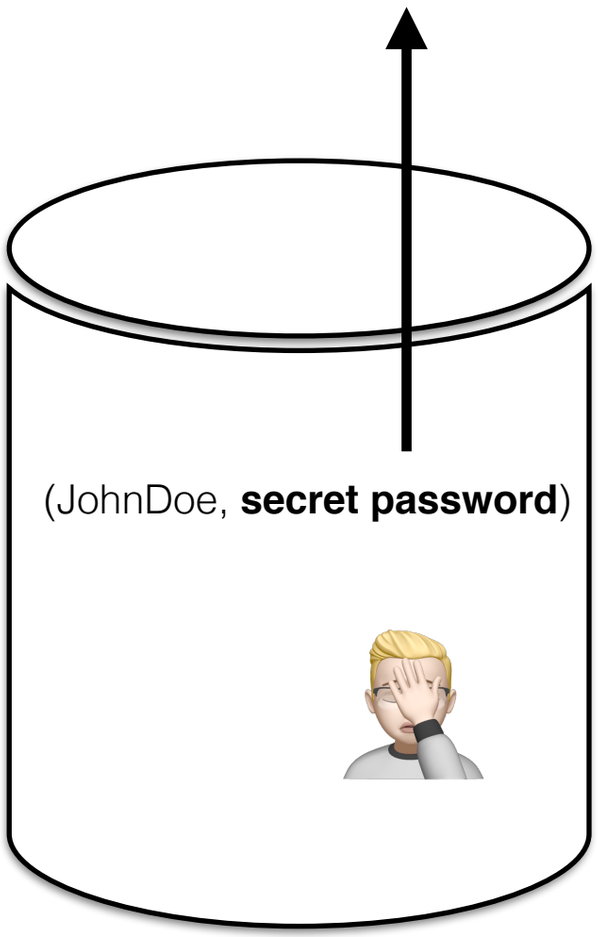
UserID: JohnDoe

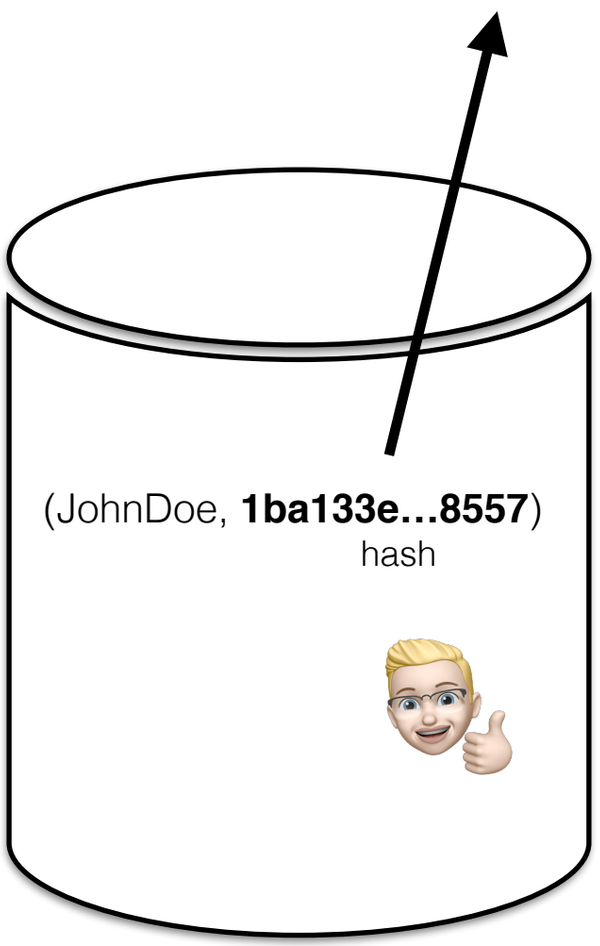
Password: **secret password**

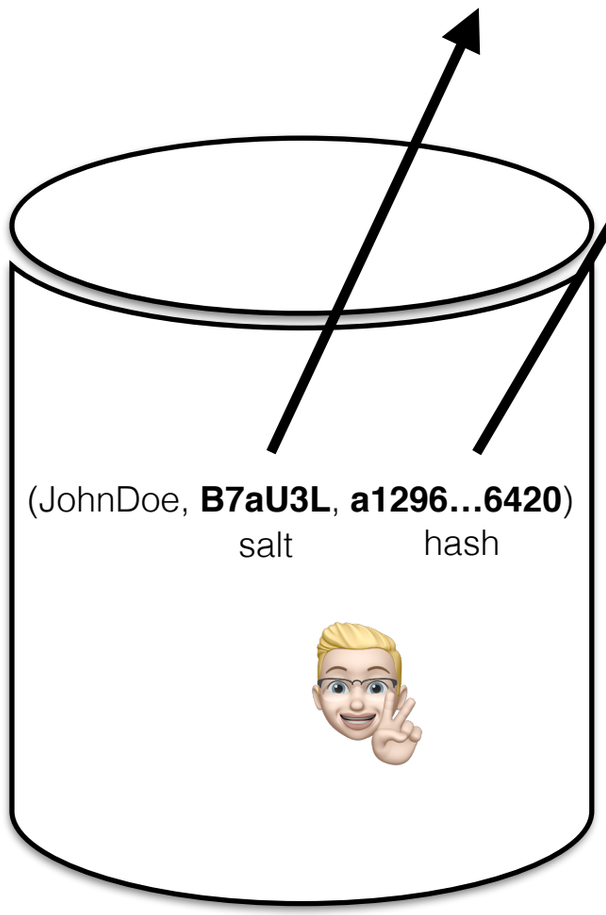
secret password == **secret password**

SHA256(**secret password)** == **1ba133e…8557**

SHA256(**secret password B7aU3L**) == **a1296…6420**







Plain text

Hashed

Salted Hash

0. Password: Plain Text

(JohnDoe, **secret password**)

Often unintentional…

Twitter: 300 million passwords in clear text (2018)

Google GSuite (2019)

100s of millions of user passwords in plain text for years (2019)

Facebook admitting having passwords in plain text (2019)

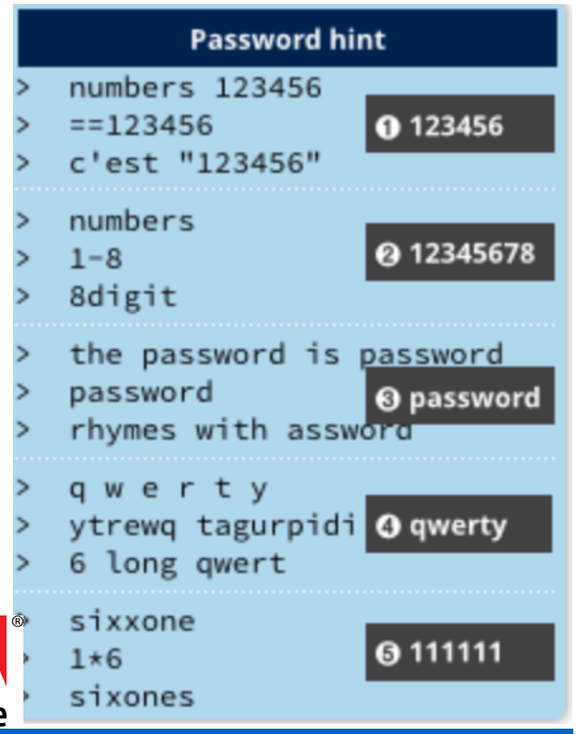
Sometimes for convenience…

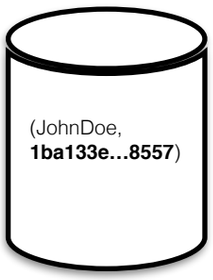
T-Mobile Australia to stop storing passwords in plain text (2018)

T-Mobile Australia: 2 million customer data hacked (2018)

Adobe leaks 150 millions passwords (2013)

Anatomy of a password disaster (2013)





1. Password: Hashed

(JohnDoe, **1ba133e…8557**)

hash

SHA-256: Cryptographic hash function converts any string into 256 bits gibberish

64 characters from {0..9a..e} (4 bits each)

"secret password" → SHA-256 → '1ba133eccdfc4e5ca3405dfd70c11360af038106c9eebdde504a4b14c94b8557'

Advantages

Deterministic: always the same answer for a given text message string

No need to ever store the password

At runtime, entered password is hashed and compared with stored hash value

Extremely low probability that two different text messages will be converted to same hash value (collision)

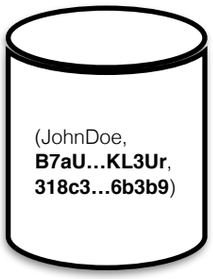
Disadvantages

Billions of passwords could be pre-hashed and stored in a lookup table (database)

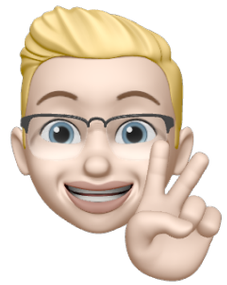
Comparing billions of hashes can be done in seconds (checking billions of passwords per second)

Relatively easy to crack if password hash is among the billions of pre-computed password hashes

Further info: rainbow tables



2. Password: Salted Hash



(JohnDoe, **B7aU…KL3Ur**, **318c3d2967274…5233bf52346b3b9**)

salt hash

Salt (cryptography): random value that is added to a password before being hashed

"secret password" + "B7aU3LS8n2KL3Ur" → SHA-256 → "318c3d2967274…5233bf52346b3b9"

salt

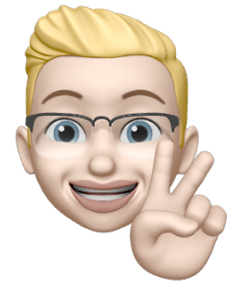
Advantages

Orders of magnitude slower to crack

Pre-computed hashed passwords cannot be used. Disables rainbow tables attacks

Necessary: **different** randomly generated **long** salt value for each user

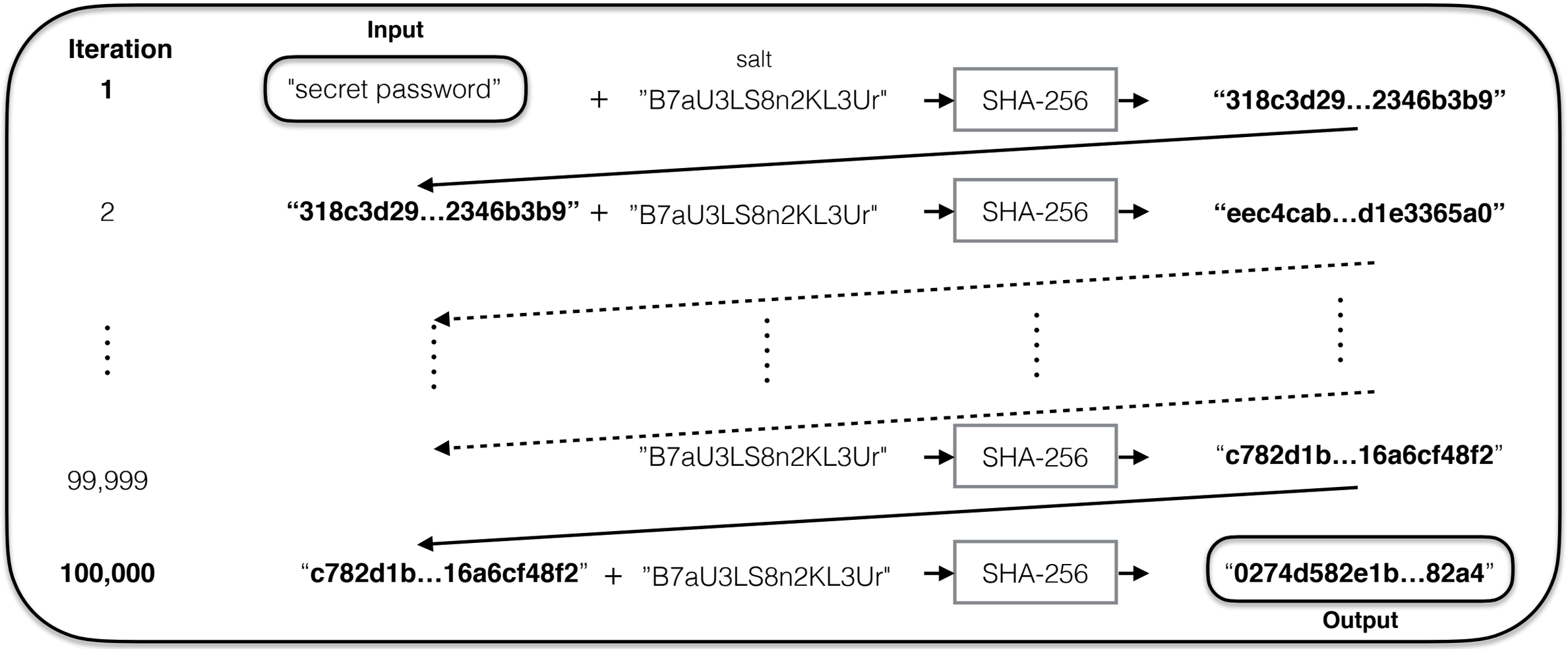
3. Password: Key stretching



(JohnDoe, **B7aU…KL3Ur**, **100,000**, **0274d582e1b…82a4**)

salt Number iterations hash

<u>Key stretching</u>: apply a hash function repeatedly in a loop (e.g. 100,000 times)



Advantages

Orders of magnitude (number of iterations) slower to crack (computational time of the attacker’s system)

Tools: Password Based Key Derivation Function 2 (<u>PBKDF2</u>), <u>Scrypt</u>, <u>Argon2</u>, <u>Bcrypt</u>, …

Password Storage (recap.)

UserID: JohnDoe

Password: **secret password**

secret password == **secret password**

SHA256(**secret password)** == **1ba133e…8557**

SHA256(**secret password B7aU3L**) == **a1296…6420**

(JohnDoe, **secret password**)

Plain text

(JohnDoe, **1ba133e…8557**)
hash

Hashed

(JohnDoe, **B7aU3L**, **a1296…6420**)
salt hash

Salted Hash

hsS8sOu…2r0lu:**hg12345678**
EEaekuiZX…G6ua:**password7**

Data Breaches

XrxigRUy…YdEVepq:**Butt3rM0nk3y**
$2a$08hW….joj5yK:**iloveyou22**

Have I Been Pwned (Troy Hunt) hashes.org

Over 9 billions pawned accounts (2019-01)

Over 500 million cracked passwords (now in plain text) (2019) (download)

Check your (old) password online: haveibeenpwned.com/Passwords

Hashing Functions (that should not be used anymore, but still are)

MD5: insecurity demonstrated in 2005 (collision attack in seconds):

In 2012: 200 million Yahoo credentials went up for sale online

In 2014: still in use at companies like Cisco, McAfee and Symantec

"**secret password**" → **"1ba133…8557"**

"some other text" → **"1ba133…8557"**

SHA-1: deprecated in 2011 by NIST, insecurity demonstrated in 2017

Data Breaches

2018-12: Quora: 100 million user accounts affected by a data breach

2018-09: Facebook: 90 million user accounts affected by a data breach

2017: Yahoo: 3 billion user accounts: names, email addresses, telephone numbers, encrypted or unencrypted security questions and answers, dates of birth, and hashed passwords.

Password Cracking

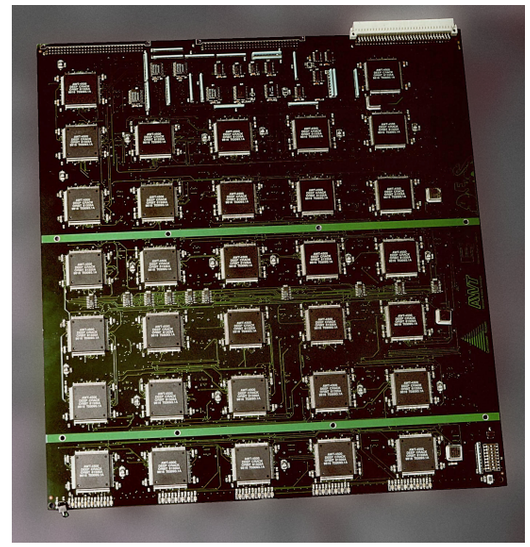
1. Brute force: Try every single possible combination of characters

2. Use of lexicons + hand crafted rules on human behaviour when creating passwords

3. Language models (deep neural networks) (to generate candidate passwords)

4. Generative Adversarial Networks (to generate candidate passwords)

1. Brute force



Use case

Feasible for cracking one given password for one given account

Feasible when the length of password is relatively small (e.g. 8 characters)

Feasible when access to huge computing power (e.g. network of computers)

Single machine: Deep Crack (1998): Cracking DES standard

10^{17} passwords space. 90 billion keys tested per second.

Maximum time needed: 9 days. Cracked in less than 3 days.

Distributed

2012: 25-GPU cluster cracks every standard Windows password in < 6 hours (Hashcat)

distributed.net: volunteers around the world donating the power of their home computers

2. Lexicon + rules

Idea: Makes use of a list of words and apply some modifications to those words

Dictionary attack

Wordlist: text string samples from a dictionary or real passwords previously cracked

Rules: variety of alterations to the dictionary words

e.g. word + append up to 3 digits

(for —> 4), (s —> 5), (i —> 1), (word —> wordDIGITS), …

“Hello” —> “Hell0”, “He11o”, “Hello123”

Tools

John the Ripper, Hashcat, Aircrack-ng

3. Language models

Step 1: Train a statistical language model to model the vast amount of cracked leaked passwords

Step 2: Use model to generate huge quantities (billions) of passwords (most of them never seen)

Step 3: Try if those passwords can match the passwords you wish to crack

Previously:

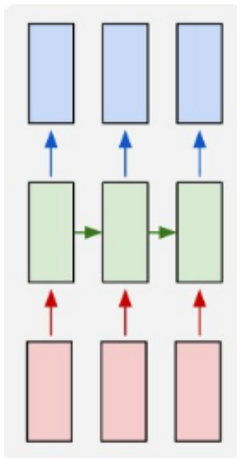
Language models based on n-grams and Markov models (e.g. Language modelling notes)

P(secret password) = P(s|<s>) P(e|<s>s) P(c|<s>se) P(r|sec) ….P(r|swo) P(d|wor)

Nowadays: Artificial Neural Networks

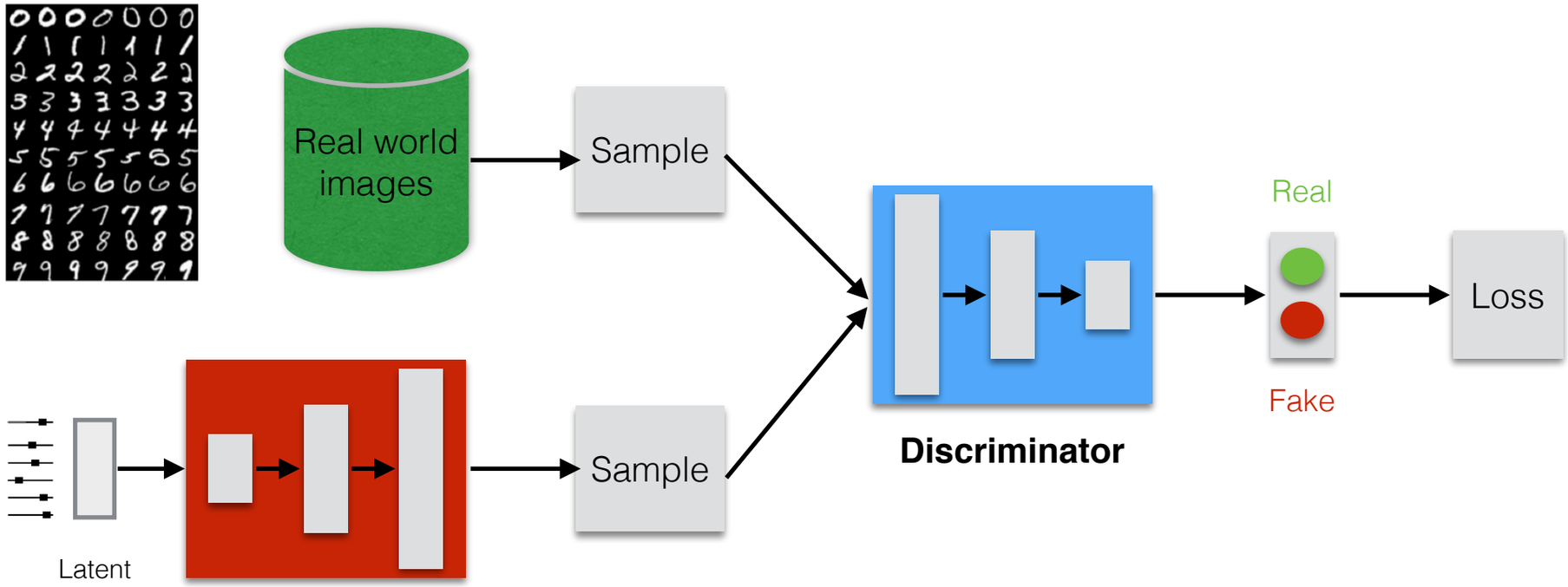
Paper: Modeling password with neural networks (2016) - Code

Tutorial: NLP from scratch: Training a character-level Recurrent Neural Networks

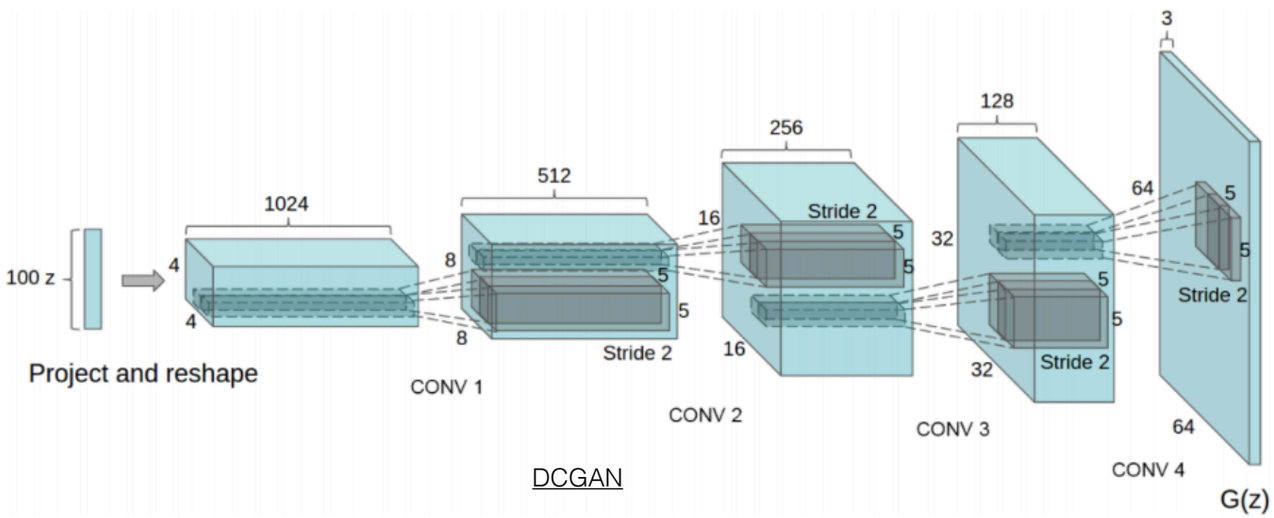


U Toronto - Lisa Zhang

4. Generative Adversarial Networks (GAN)



didier.guillevic.net

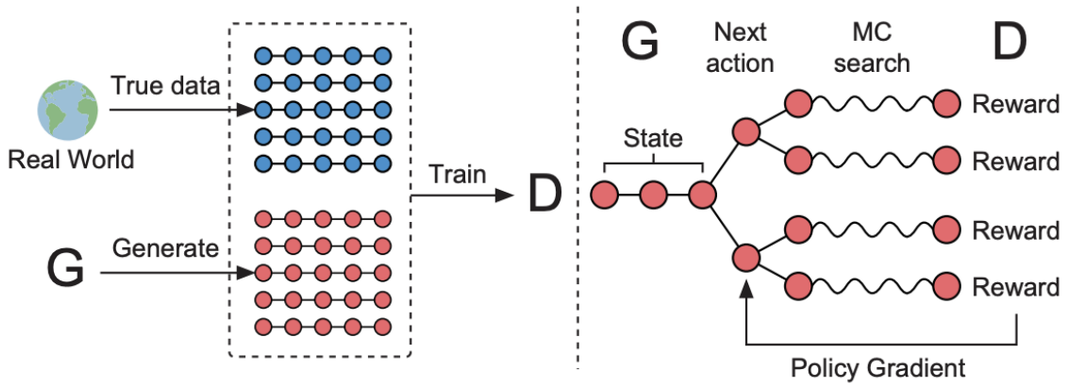


DCGAN

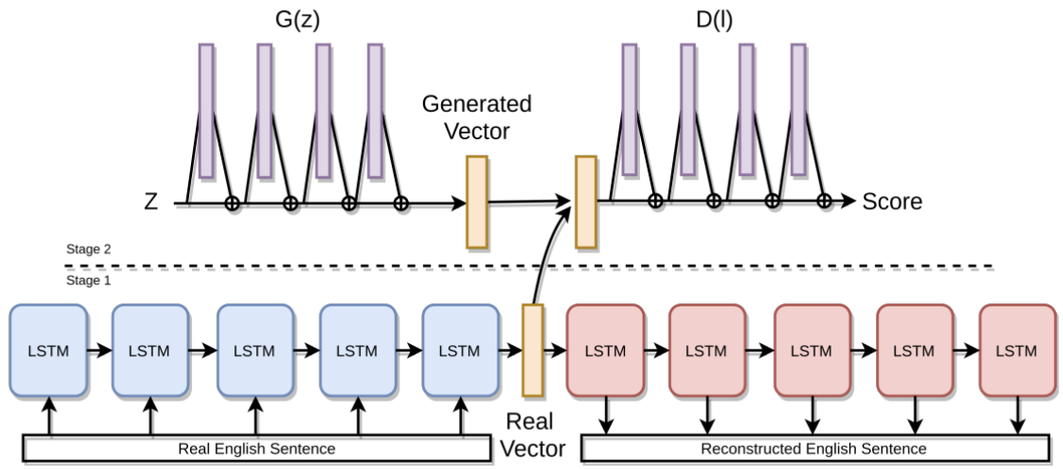
Progressive GAN
10/2017
1024 x 1024

Progressive GAN

4. Generative Adversarial Networks (GAN) (cont.)



SeqGAN (2017-08) (Reinforcement Learning)



Adversarial Text Generation Without Reinforcement Learning (2019-01)

PassGAN: A Deep Learning Approach for Password Guessing (2019) (code)

Used to augment / supplement dictionary tools like John The Ripper and Hashcat

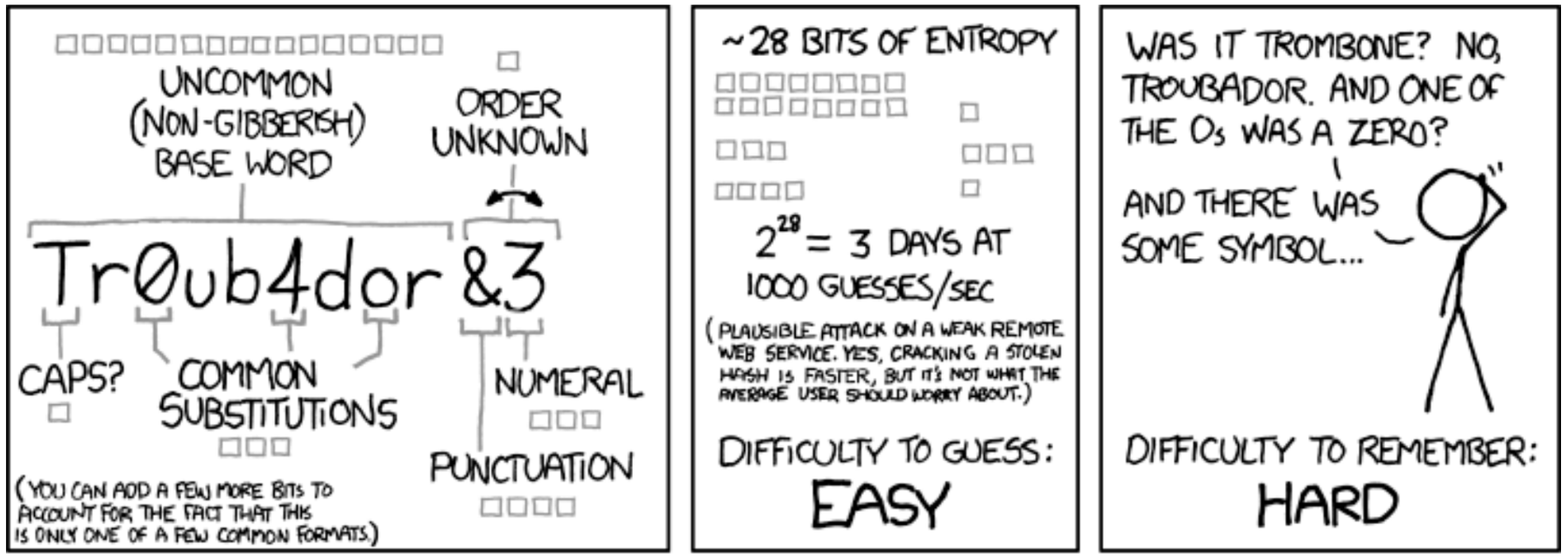
Use model to generate billions of passwords

Toward better passwords

10^6 words * 10^3 variations

10^9 passwords

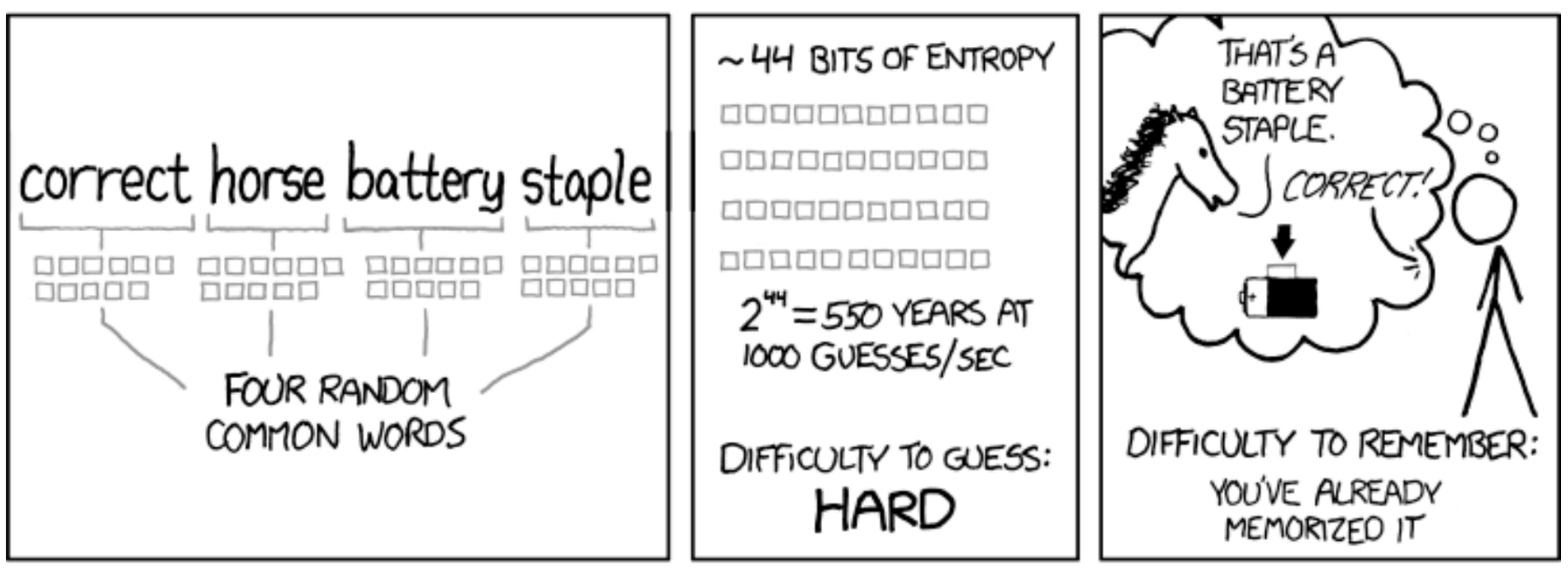
!!! 1 second !!! to crack with precomputed hash values



$(10^5$ words$)^4$

10^{20} passwords

!!! 3000 years !!! to crack with precomputed hash values



xkcd.com

Toward better passwords (cont.)

Most frequent passwords: 123456, 123456789, qwerty, password, 111111

Password strength: a function of length, complexity, and unpredictability

1. **Long**: Absolutely needs to be longer than 8 characters (long passwords)
2. **Unpredictable**: think information entropy

Four to five "random" words

dreamstime.com

submarine-ballon-cloud-goeland

(not 4 words that are present on your public social media page; e.g. LinkedIn profile)

Roll a dice to avoid predictability and select 4 words from a list of predefined words

Arnold Reinhold (1995): Diceware, (1Password blog)

Think passphrase rather than password

Video: Edward Snowden on Passwords (2015) (John Oliver Show) <— To watch for a good laugh!

Password Management Tool

BitWarden.com (open source, desktop, mobile, browser plugins)

NordPass.com (freemium, desktop, mobile, browser plugins)

1password.com (commercial, desktop, mobile, browser plugins)

LastPass.com (commercial, desktop, mobile, browser plugins)

KeePass.info (open source)

Office of the Privacy Commissioner of Canada
Commissariat à la protection de la vie privée du Canada

TIPS FOR CHOOSING, USING AND PROTECTING PASSWORDS

If passwd mgt tool: find one you think you can trust

Multi-factor authentication

Authentication factors (Wikipedia)

Something you have: physical object: bank card, key, USB stick, …

Something you know: password, PIN, , …

Something you are: physical characteristics of the user (biometrics): fingerprint, eye, voice, typing speed, …,

Somewhere you are: e.g. GPS signal to identify the location, specific computing network connection

Two Factor Identification

SMS

(2018) SMS: dominant multi-factor authentication method for consumer-facing accounts

Susceptible to phone number being stolen (ported) through social engineering

One time generator passwords

RSA SecurID (2003): token assigned to a computer user: generate one-time passwords

(2011) System compromised (cyber attack) originating from phishing emails with Excel file

Google Authenticator: mobile app: Time based One-Time Password (TOTP) algorithm

FreeOTP (RedHat): mobile app: TOTP, HMAC-based One-time Password (HOTP) algorithm

Authy: mobile app

Security key - Fast IDentity Online (FIDO) Alliance

YubiKey: USB stick, Bluetooth

TouchID, FaceID open to this party developers: iOS apps can employ FIDO authentication

Resources

Tools: John the Ripper, Hashcat, Aircrack-ng, Language Model (Neural), Language Model (GAN)

Leaked password lists: hashes.org, Openwall wordlists collection

Advice

1. Use a different password for each account / web site
2. Use a passphrase instead of a password (Better passwords, Information entropy)
3. Use a password management tool (e.g. 1Password (commercial), Bitwarden (open source), NordPass)
4. Use two factor identification (e.g. FreeOTP, Authy, Google Authenticator)

Videos: Edward Snowden on passwords (<— Highly recommended :-))

Bonus - Getting familiar with entropy (information theory)

Entropy: measure of how surprised we are about the next character or word

Chris Olah's post (highly recommended): Visual Information Theory

Better Master Passwords: The geek edition (1Password)