School of

Engineering and Computer Science

Te Kura Mātai Pūkaha, Pūrorohiko

CYBR 171 T1 2023

Ngā whakapūtanga o Te Haumaru rorohiko
Cybersecurity Fundamentals

Authentication I

Passwords and Tokens (*hardware*)

Learning goals

- Passwords
  - How to meet conflicting requirements
  - Hashing
  - Salts
- What's wrong with passwords and some fixes
- How **hardware tokens** can be classified in terms of types of passwords and connectivity.
- Outline how **TOTP** and **Challenge-Response** password types work.
- Identify hardware token vulnerabilities.

PART I:
Keeping Passwords Secret

Ways to authenticate users

The four means of authenticating user identity are based on:

Something the individual **knows**	Something the individual **possesses** (token)	Something the individual **is** (static biometrics)	Something the individual **does** (dynamic biometrics)
• Password, PIN, answers to prearranged questions	• Smartcard, electronic keycard, physical key	• Fingerprint, retina, face	• Voice pattern, handwriting, typing rhythm

Password authentication

- **Widely used** line of defense against intruders
  - The user **provides name/login** and **password**
  - The system **compares** the password with the one stored for that specified login
- The user ID:
  - **Determines** that the user is **authorized** to access the system
  - **Determines** the user's **privileges**
- Passwords stored in a password file ("passwd" in Unix)
- Operating system will provide some **protection**:
  - Limit on how many **times** can guess password
  - Check **origin** of login ("is Ian really in China this week?")
  - "passwd" file might be **hidden** from direct access except by root

Online versus Offline attacks

- **Online**
  - Attacker keeps trying passwords until gets access
  - **Brute force** tries every combination
  - **Dictionary** attacks use lists of passwords to increase chances of success
  - Protection system will **limit the rate** of attack
- **Offline**
  - Attacker steals the password file
  - Runs brute force or dictionary at leisure or,
  - **Exploit flaw** in implementation of password file

Conflicting requirements

1. When the user logs in must be **possible** to check have entered the correct password.
2. Even if the **database leaked** and an attacker has a huge amount of **computing power**:
   a. Password file should not reveal **repeated patterns**.
   b. Users have several days/weeks to **change password**.
3. Cannot **recover** a forgotten password even if you are a system administrator (root).

#1 Hash the password

- Remember hashes?
  - Trapdoor or **one-way** function
  - Can't determine input without brute force attack
- Password hash in password file indexed by user
- Compute the hash of the entered password
- Matching hashes mean you have entered the right password
- **Do these meet our requirements?**

Evaluation of hashing the password

- **Password is never stored**
  - Possible to check for the correct password
  - No one can recover a forgotten password
- **Hashing is deterministic**
  - HASH(“A”) always equal to HASH(“A”)
  - Password file **will** reveal repeated patterns
- SHA-256 is a common algorithm
  - 8-character password brute estimated safe for 27 days
  - **Rainbow tables** reduce this considerably
  - Users **may not** have several days/weeks to change their password

Rainbow tables

- A ***rainbow table*** is a **dictionary** of hashes
  - One table for each hash algorithm and input length
  - Hash value to original input value
  - Stored in a data structure optimised for fast lookups
- The attacker reads the password file
  - Looks up original value in the dictionary
  - That's it.
- Rainbow tables are very large, but an online service exists: https://crackstation.net/ (blocked at Uni)

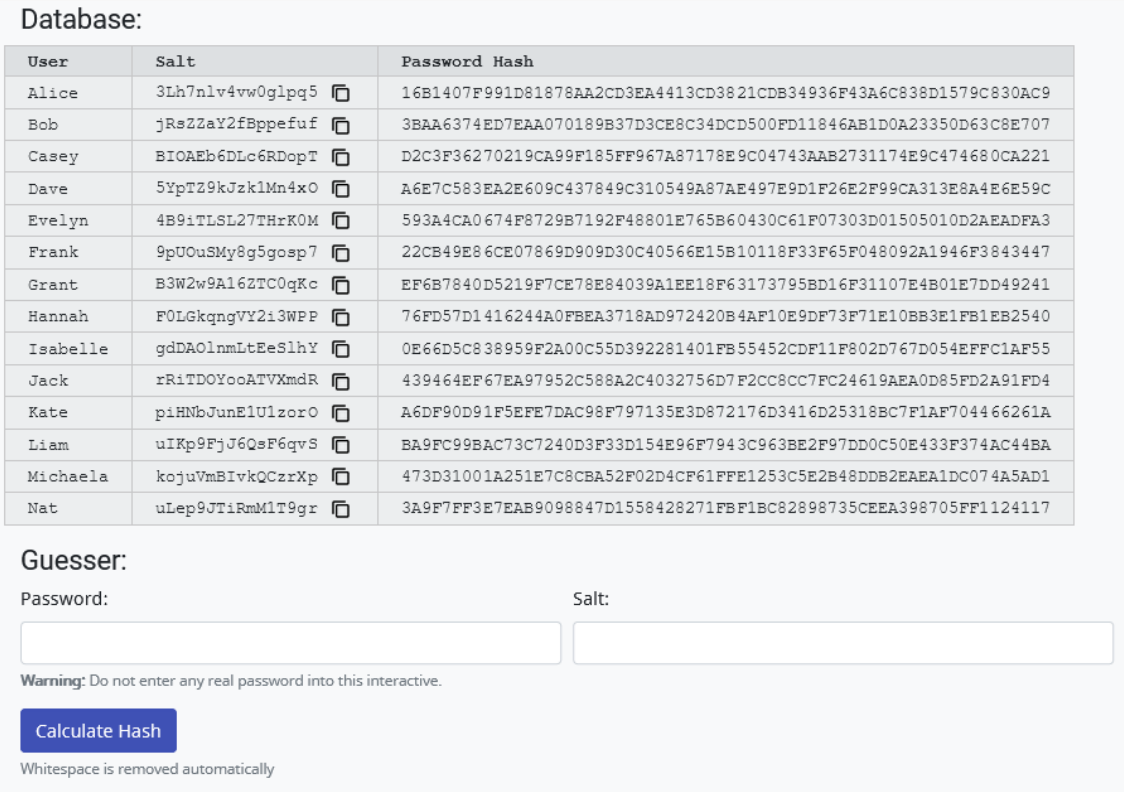
#2 Hashing the password with a salt

- Key idea:
  - Pick a **random number** (“salt”)
  - Attach to the **end** of the password
  - Hash **the combined** salt and password
  - Store salt in the password file indexed by the user
- User enter their password
  - The system retrieves password and salt
  - Hash combined password with user’s salt
  - Hash entered password with user’s salt

Evaluating hashing password with salt

- Sounds **counterintuitive** to store the salt?
- Brute force or dictionary password guessing harder
  - Still need to try all the passwords or most likely ones
  - The attacker has extra work in combining password and salt
- Rainbow attacks are made **extremely hard**
  - Table for each salt value
  - Common salt length 128 bits
  - Approx. 3.4×10^{38} tables
- ***All of our requirements are met by salting the password***
- Side note – a closely related approach known as **stretching** the password

What this looks like in practice



Database:

User	Salt	Password Hash
Alice	3Lh7nlv4vw0glpq5	16B1407F991D81878AA2CD3EA4413CD3821CDB34936F43A6C838D1579C830AC9
Bob	jRsZZaY2fBppefuf	3BAA6374ED7EAA070189B37D3CE8C34DCD500FD11846AB1D0A23350D63C8E707
Casey	BIOAEb6DLc6RDopT	D2C3F36270219CA99F185FF967A87178E9C04743AAB2731174E9C474680CA221
Dave	5YpTZ9kJzk1Mn4xO	A6E7C583EA2E609C437849C310549A87AE497E9D1F26E2F99CA313E8A4E6E59C
Evelyn	4B9iTLSL27THrK0M	593A4CA0674F8729B7192F48801E765B60430C61F07303D01505010D2AEADFA3
Frank	9pUOuSMy8g5gosp7	22CB49E86CE07869D909D30C40566E15B10118F33F65F048092A1946F3843447
Grant	B3W2w9A16ZTC0qKc	EF6B7840D5219F7CE78E84039A1EE18F63173795BD16F31107E4B01E7DD49241
Hannah	F0LGkqngVY2i3WPP	76FD57D1416244A0FBEA3718AD972420B4AF10E9DF73F71E10BB3E1FB1EB2540
Isabelle	gdDAOlnmLtEeSlhY	0E66D5C838959F2A00C55D392281401FB55452CDF11F802D767D054EFFC1AF55
Jack	rRiTDOYooATVXmdR	439464EF67EA97952C588A2C4032756D7F2CC8CC7FC24619AEA0D85FD2A91FD4
Kate	piHNbJunE1U1zorO	A6DF90D91F5EFE7DAC98F797135E3D872176D3416D25318BC7F1AF704466261A
Liam	uIKp9FjJ6QsF6qvS	BA9FC99BAC73C7240D3F33D154E96F7943C963BE2F97DD0C50E433F374AC44BA
Michaela	kojuVmBIvkQCzrXp	473D31001A251E7C8CBA52F02D4CF61FFE1253C5E2B48DDB2EAEA1DC074A5AD1
Nat	uLep9JTiRmM1T9gr	3A9F7FF3E7EAB9098847D1558428271FBF1BC82898735CEEA398705FF1124117

Guesser:

Password:

Salt:

Warning: Do not enter any real password into this interactive.

Calculate Hash

Whitespace is removed automatically

https://www.csfieldguide.org.nz/en/interactives/password-guesser/

PART II:

What's wrong with passwords (and some fixes)

Default Password

- Many devices ship with simple publicly-documented usernames and passwords.
- Attackers have exploited Internet connected-devices.
- Launch other attacks on other computers.
  - Mirai (2016)
  - 380,000+ devices
- Breach of confidentiality.
  - Webcams and baby monitors

Stranger hacks couple's video baby monitor to watch baby

Mum heard voice in the night saying, 'Wake up baby!'

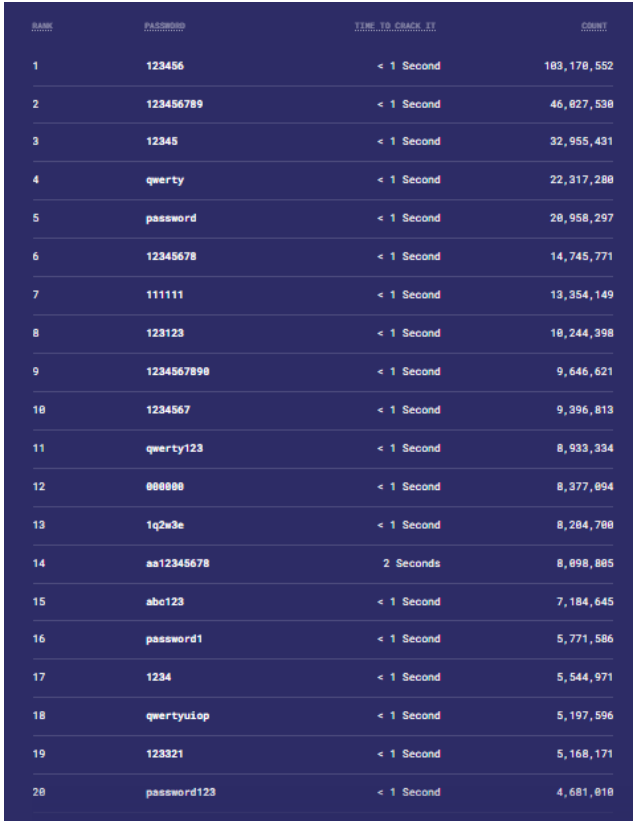
By Jessica Gibb

Countermeasures

- Users
  - **Change default passwords**
  - Interfaces are not intuitive in most cases
- Manufacturers
  - Use **unique** default passwords
    - E.g. Put a sticker with a unique password on the router
  - Use **alternative** authentication mechanisms
    - E.g. Press Wi-Fi Protected Setup (WPS) button
  - **Force** password change
    - E.g. When connect the first time you must set a new password
  - **Restrict** network access
    - E.g. Only this one computer has access to the device

Weak Password

- Weak passwords
  - Speed by brute forcing
  - Length and complexity
- Categories
  - Short passwords
  - Dictionary words
  - Common passwords
- Example
  - https://nordpass.com/most-common-passwords-list/



RANK	PASSWORD	TIME TO CRACK IT	COUNT
1	123456	< 1 Second	103,170,552
2	123456789	< 1 Second	46,827,538
3	12345	< 1 Second	32,955,431
4	qwerty	< 1 Second	22,317,288
5	password	< 1 Second	20,958,297
6	12345678	< 1 Second	14,745,771
7	111111	< 1 Second	13,354,149
8	123123	< 1 Second	10,244,398
9	1234567890	< 1 Second	9,646,621
10	1234567	< 1 Second	9,396,813
11	qwerty123	< 1 Second	8,933,334
12	000000	< 1 Second	8,377,094
13	1q2w3e	< 1 Second	8,284,700
14	aa12345678	2 Seconds	8,098,805
15	abc123	< 1 Second	7,184,645
16	password1	< 1 Second	5,771,586
17	1234	< 1 Second	5,544,971
18	qwertyuiop	< 1 Second	5,197,596
19	123321	< 1 Second	5,168,171
20	password123	< 1 Second	4,681,010

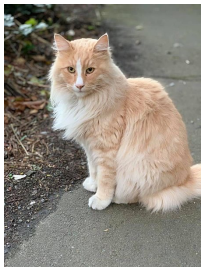
Countermeasure - *Complexity*

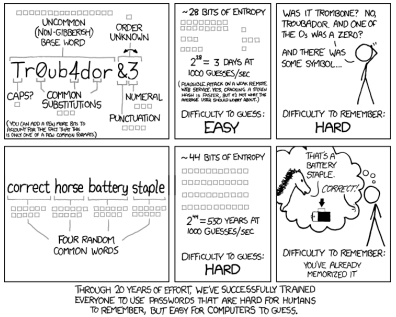
- Choose better passwords
- Complexity requirements
  - (1) minimum length and (2) range of acceptable characters
  - Examples:
    - Windows: 6+ characters, 3 of 4 categories: uppercase, lowercase, 0-9 and symbols.
    - Apple: 8+ characters, at least 1 number, 1 uppercase letter and 1 lowercase letter.
- Password hints or reset questions
  - People will forget complex passwords
  - Allow hints or additional verification questions to reset password

Better Countermeasure - *Passphrases*

- Complexity requirements
  - Hard to remember so write the password down
  - Choose easy to answer or guess hints or verification questions i.e. what is the name of your pet?
- NIST (2017) standards
  - Use a passphrase i.e. catstaplesbluedog
  - Make it long 8-64 characters to slow guessing
  - xkcd web comic

    [xkcd: Password Strength]

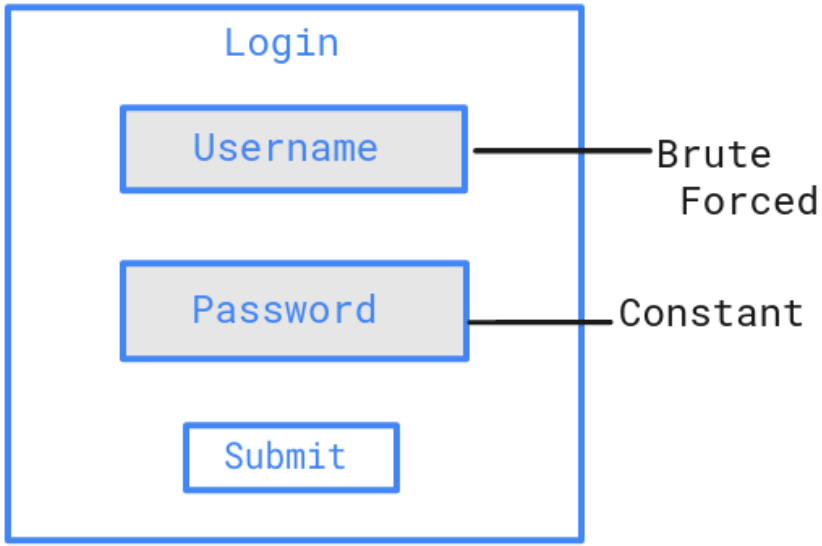




Password Spraying

1. Attacker obtains a list of usernames on site.
2. Use **same password** on **many different accounts**.

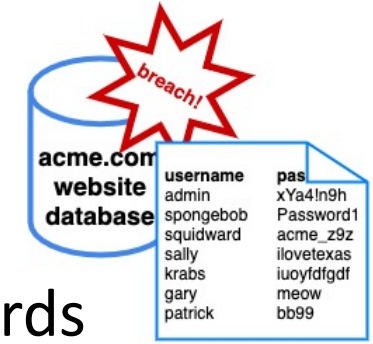
Avoids **lockout** from too many tries on one account, need to lockout if the same machine tries too many times across ALL accounts.

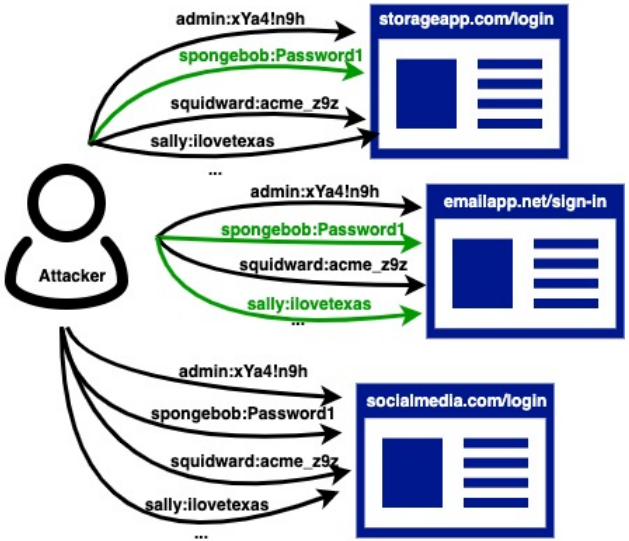


Credential stuffing

Credential is username and password combination

1. Attacker acquires usernames and passwords (data breach/password dump).
2. Attacker uses automated tools to test stolen credentials against many websites.
3. If the login is successful, the attacker knows they have a set of valid credentials.





Countermeasure – *Password per site*

- Use passphrases instead of passwords
  - Indiana University study (2018)
  - Passphrase requirement 15+ characters
  - Found 98.98 percent of users did not appear in breaches (i.e. reuse was low)
- Use a personalised algorithm
  - Choose a good password
  - For each site, combine the site name with the password
  - You must do it in a non-obvious way similar to hash
  - It might be hard to apply on a regular basis

Countermeasure – Password per site (cont.)

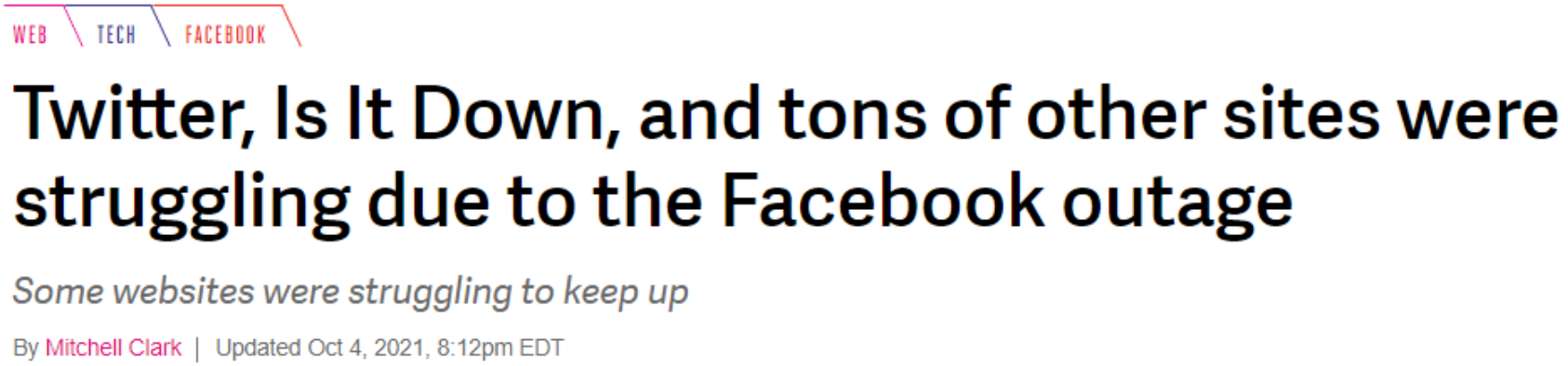
- Use a **browser-based password manager**
  - Limits you to that particular browser
  - Must synchronise across different machines
  - E.g. https://support.mozilla.org/en-US/kb/password-manager-remember-delete-edit-logins
- Use an **online password manager**
  - Examples: Lastpass, dashlane, dropbox
  - Master password to access via the web
  - Browser and mobile phone applications
  - Centralised database attractive to hackers
    - LastPass says no passwords were compromised following breach scare - The Verge

Countermeasure – *Single Sign On (SSO)*

- Use one password everywhere for several related but independent software systems
  - Simplifies burden on users and administrators
  - Must make sure it is a very good password
  - User doesn't reveal password to independent systems
- Examples:
  - Kerberos for corporate environments
  - Oauth for web services
    - Open Authorization standard
    - Authentication and access to end user account information
    - Google, Facebook, Microsoft, Apple …

Risks associated with OAuth

- What happens when the Oauth authentication provider goes down?



WEB \ TECH \ FACEBOOK

Twitter, Is It Down, and tons of other sites were struggling due to the Facebook outage

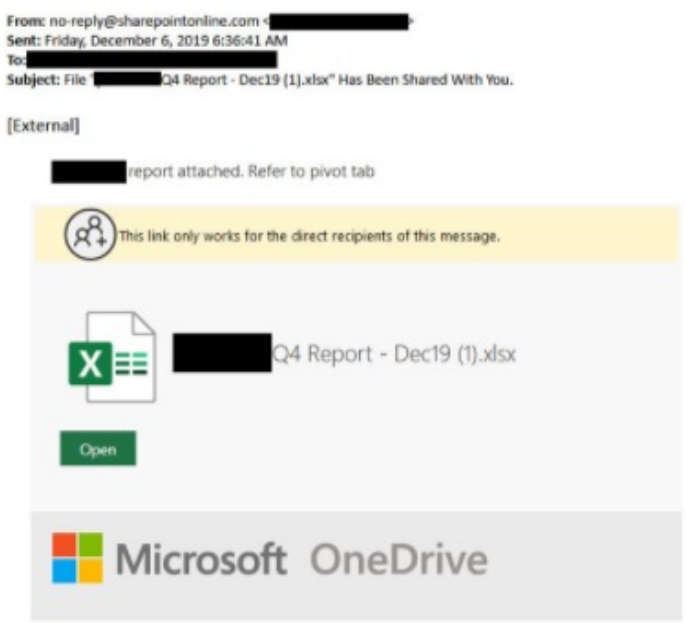
Some websites were struggling to keep up

By Mitchell Clark | Updated Oct 4, 2021, 8:12pm EDT

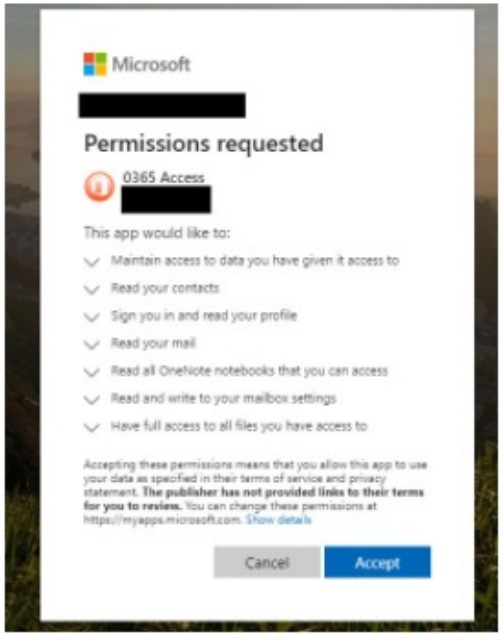
https://www.theverge.com/2021/10/4/22709123/facebook-outage-down-detector-cloudflare

Risks associated with Oauth (cont.)

- What happens when you give away too much access to a rogue application?



Phishing email sample



Office 365 OAuth app

https://www.bleepingcomputer.com/news/security/microsoft-warns-of-increasing-oauth-office-365-phishing-attacks

PART III:
Hardware tokens

Ways to authenticate users

The four means of authenticating user identity are based on:

Something the individual **knows**

- Password, PIN, answers to prearranged questions

Something the individual **possesses** (token)

- Smartcard, electronic keycard, physical key

Something the individual **is** (static biometrics)

- Fingerprint, retina, face

Something the individual **does** (dynamic biometrics)

- Voice pattern, handwriting, typing rhythm

Security (*hardware*) tokens

Raised edges push pins above cylinder

Memory and processor, radio antenna or gold connector

No memory, magnetic strip read by door lock

Memory and processor, user enters PIN on the display

Ways to classify (*hardware*) tokens

- Password type
  - Fixed
  - Dynamic
    - One time password
    - Challenge-response

- Connectivity
  - Disconnected
  - Contact
  - Contactless

Password types - *Fixed*

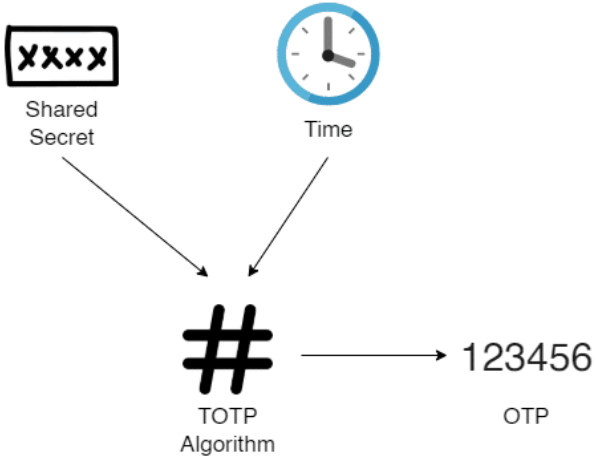
- Fixed
  - **Never changes**
  - The attacker can replay to gain access ("**replay attack**")
- Example
  - The card stores a **secret password**
  - The authentication server checks the password
  - Attackers copy the card contents
  - Attacker uses copied card for access

Password types - *Dynamic*

- Dynamic
  - **Changes regularly**
  - The attacker **cannot** use a replay attack
- **Two** main types
  - **One-time password** (OTP)
    - Hash-based OTP (synchronised **counter**)
    - Time-based OTP (synchronised **time**)
  - **Challenge-response**
    - One party poses a **question**: "**challenge**"
    - Other party must give a valid **answer**: "**response**"
    - These challenges and responses change every time

Time-based One-Time Password

- Token and server have synchronised clocks
- Shared secret key
- Cryptographic hash algorithm
- Fixed-size output easily converted to a string of digits
- Password **never repeated** and **changes** at regular intervals



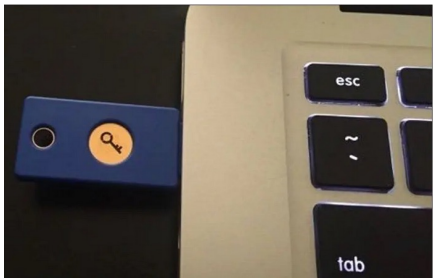
https://rublon.com/blog/what-is-totp/

Cryptographic Challenge-Response

- Token has a **secret key**
- The server has **matching public key**
- **Challenge** sent to the **token**
  - Random number (nonce)
  - Encrypted using the token's public key
- **Response** sent to the **server**
  - Nonce value + 1
  - Encrypted using the token's private key
- Only a valid token could generate the correct response

Connectivity

- Disconnected
  - **Displays** password on screen
- Contact
  - Physically via USB port
  - Audio jack
  - Inserted into a reader
- Contactless
  - Near field communication (NFC)
  - Bluetooth
  - Visual (QR code shown to camera)
  - Out-of-band - SMS



Vulnerabilities

- **Loss and theft**
  - Use someone else's token
- **Predict the one-time password**
  - Find out the secret key
  - Run the clock in reverse so can reuse a password
- **Predict the cryptographic challenge-response**
  - Extract the private key
- **Forced downgrade**
  - Attacker prevents token from communicating with the server
  - Fall back to an **insecure** second method of authentication such as credit card signatures

Example: Extract the private key

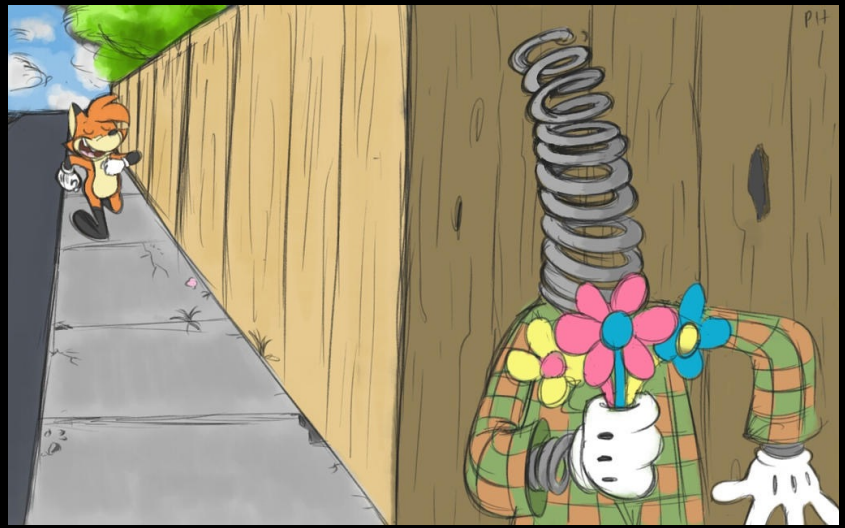
- Victor Lomne and Thomas Roche (Jan 2021)
- Attack on the **Google Titan key**
- Key contains a private key used for challenge-response
- **Monitor radio emissions** thousands of challenges
- Clone the Google Titan key
- Return the original to the target of attack

Should we no longer trust the technology?

Example: Maybe its just too much work

- **Step one** – setup measuring equipment
  - Buy $10,000 piece of equipment
  - Adjust to within one-hundredth of a millimetre
- **Step two** – open up titan key
  - Heat gun destroyed outer casing
- **Step three** – remove protection around chip
  - Fuming nitric acid
  - Highly flammable and used for rocket fuel
- **Step four** – collect measurements
  - About 6,000 runs of the challenge response protocol
- **Step five** – work out the private key
  - Lots of statistics and deep learning computations

PART IV:
What’s next



What's up next

- Assessments
  - Lab 2 was out this morning, and you have two weeks to complete it
  - Assignment 1 you still have three weeks to complete it

- Tomorrow, we will look at software tokens and biometrics.