Authentication

Applied Information Security
Summer 2021, Lecture 6

Methods of Authentication

you are characterized by a (large) set of attributes.

identity: set of attributes. you have many identities (citizen, student, ...)
enrollment: validate your attributes, before your identity added to system.
authentication: “given an identity, and some attributes, do these match?”

inconvenient to do enrollment-level validation (physical presence, interview, etc.) when you log in. instead, check fewer, **easy to protect & hard to spoof** attributes:

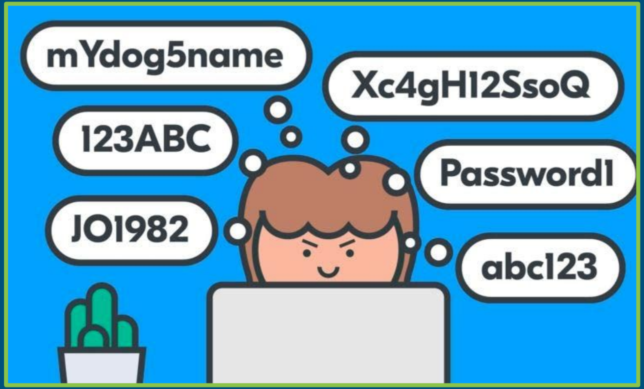
something you **know**
something you **have**
something you **are**

methods, or **factors**, of authentication

now: modern, state of the art, of authentication.

something you know:

passwords



you heard about
password policy.
how do you make a good password that satisfies it?

Strong password that’s easy to remember

Bruce Schneier’s password scheme:

1. choose a **personal sentence**.
2. combine it with some **personal tricks**
   so that it modifies this sentence to create a robust password.

easy to remember

hard to reverse

this is, effectively, a hash function

ex: “When I was in Grade 4, I forged my dad’s signature” → W1wiG4,,,1fmds)

tricks used here: initial letters ; I → 1 ; , → ,,, ;) at the end

Strong password that’s easy to remember

Bruce Schneier’s password scheme:

1. choose a **personal sentence**.
2. combine it with some **personal tricks**

   so that it modifies this sentence to create a robust password.

easy to remember

hard to reverse

this is, effectively, a hash function

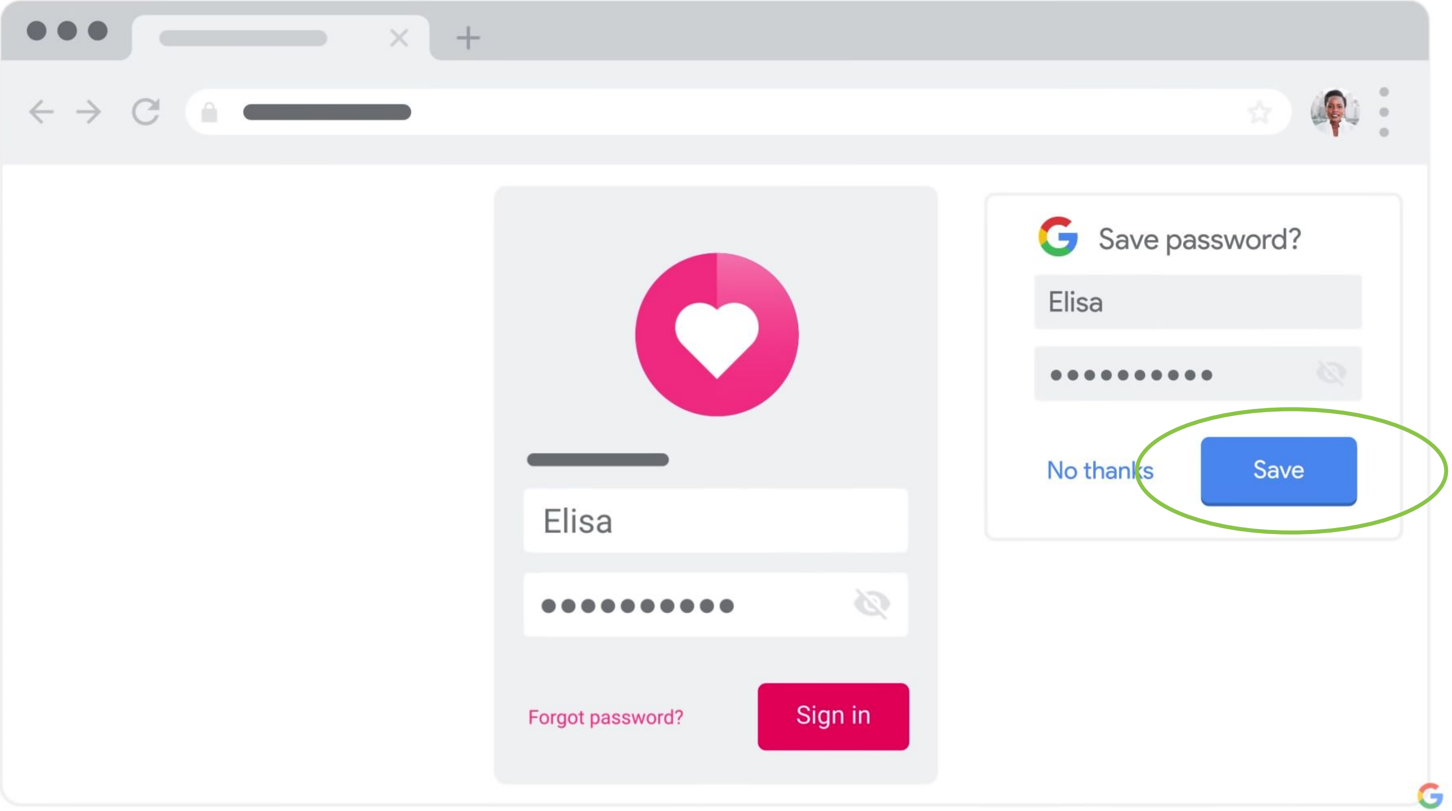
ex: “When I was in Grade 4, I forged my dad’s signature” → W1wiG4,,,1fmds)

tricks used here: initial letters ; I → 1 ; , → ,,, ;) at the end

problem: re-used scheme is inferable. remembering many schemes is hard.
solution: use this password as a master password in a **password manager**.

passwords - password manager

Save Password



Elisa

●●●●●●●●●●

Forgot password?

Sign in

Save password?

Elisa

●●●●●●●●●●

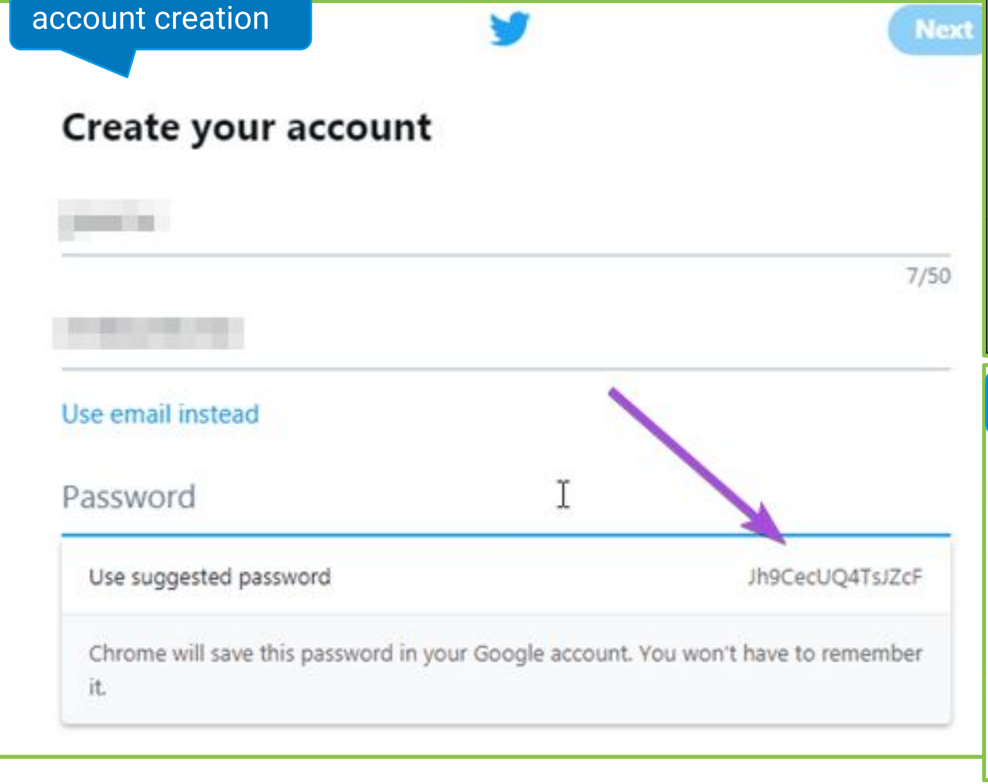
No thanks

Save

passwords - password manager

Generate Password

account creation



Next

Create your account

[redacted]

7/50

[redacted]

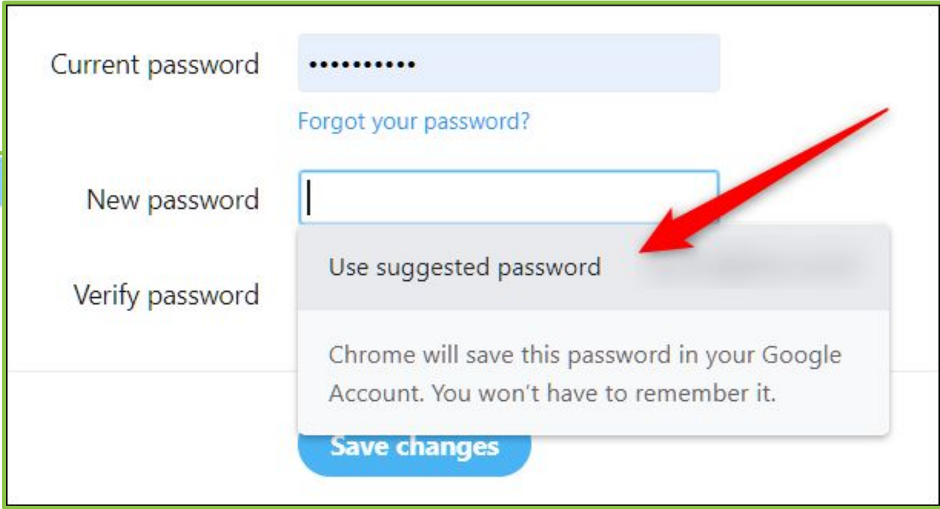
Use email instead

Password

Use suggested password   Jh9CecUQ4TsJZcF

Chrome will save this password in your Google account. You won't have to remember it.

password reset



Current password ••••••••••

Forgot your password?

New password

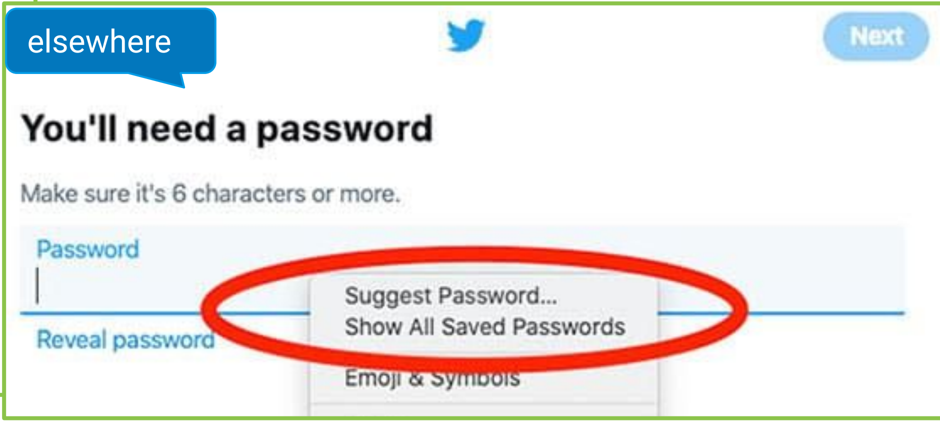
Verify password

Use suggested password

Chrome will save this password in your Google Account. You won't have to remember it.

Save changes

elsewhere



Next

You'll need a password

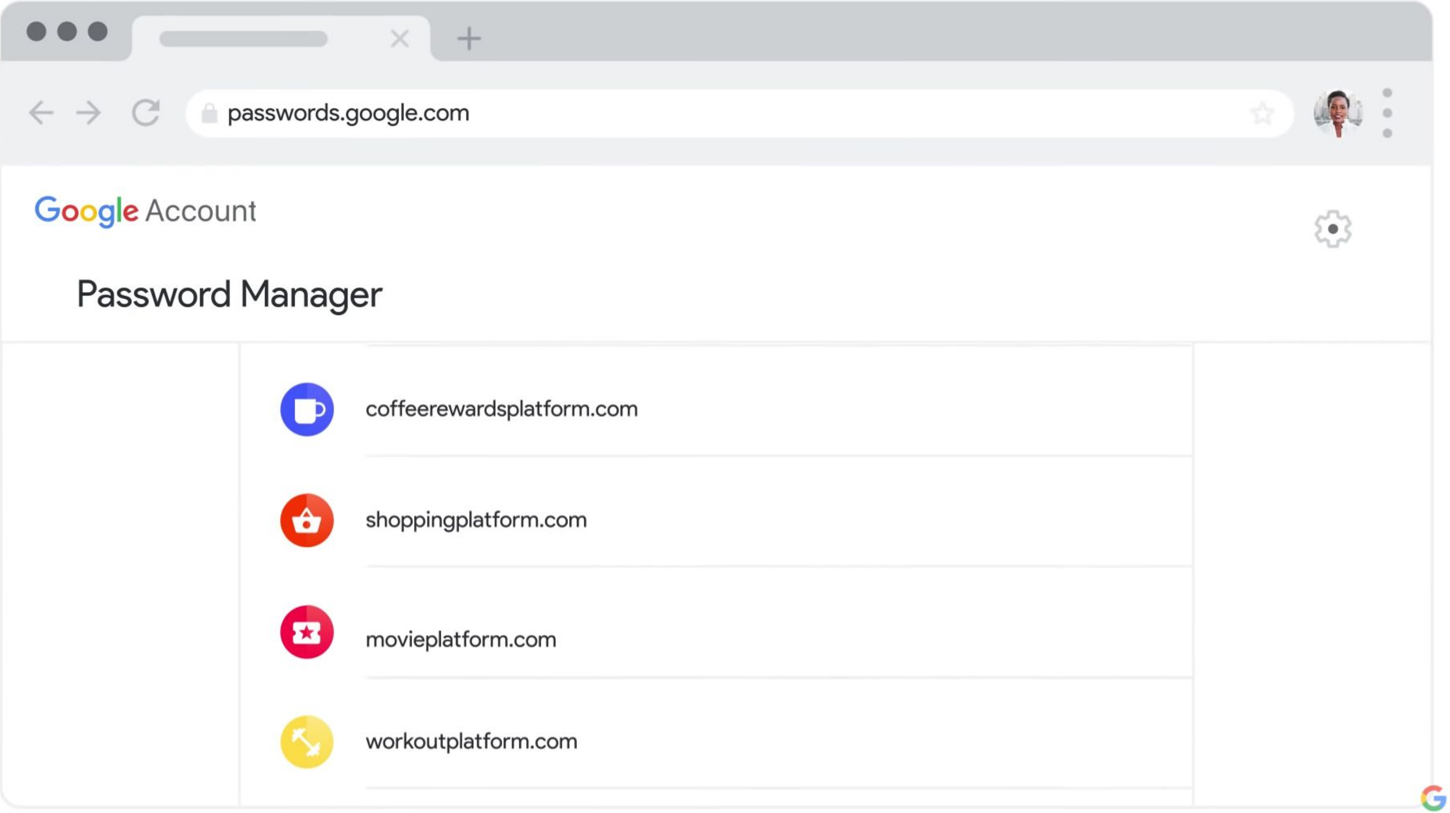
Make sure it's 6 characters or more.

Password

Reveal password

passwords - password manager

View Passwords



passwords.google.com

Google Account

Password Manager

coffeerewardsplatform.com

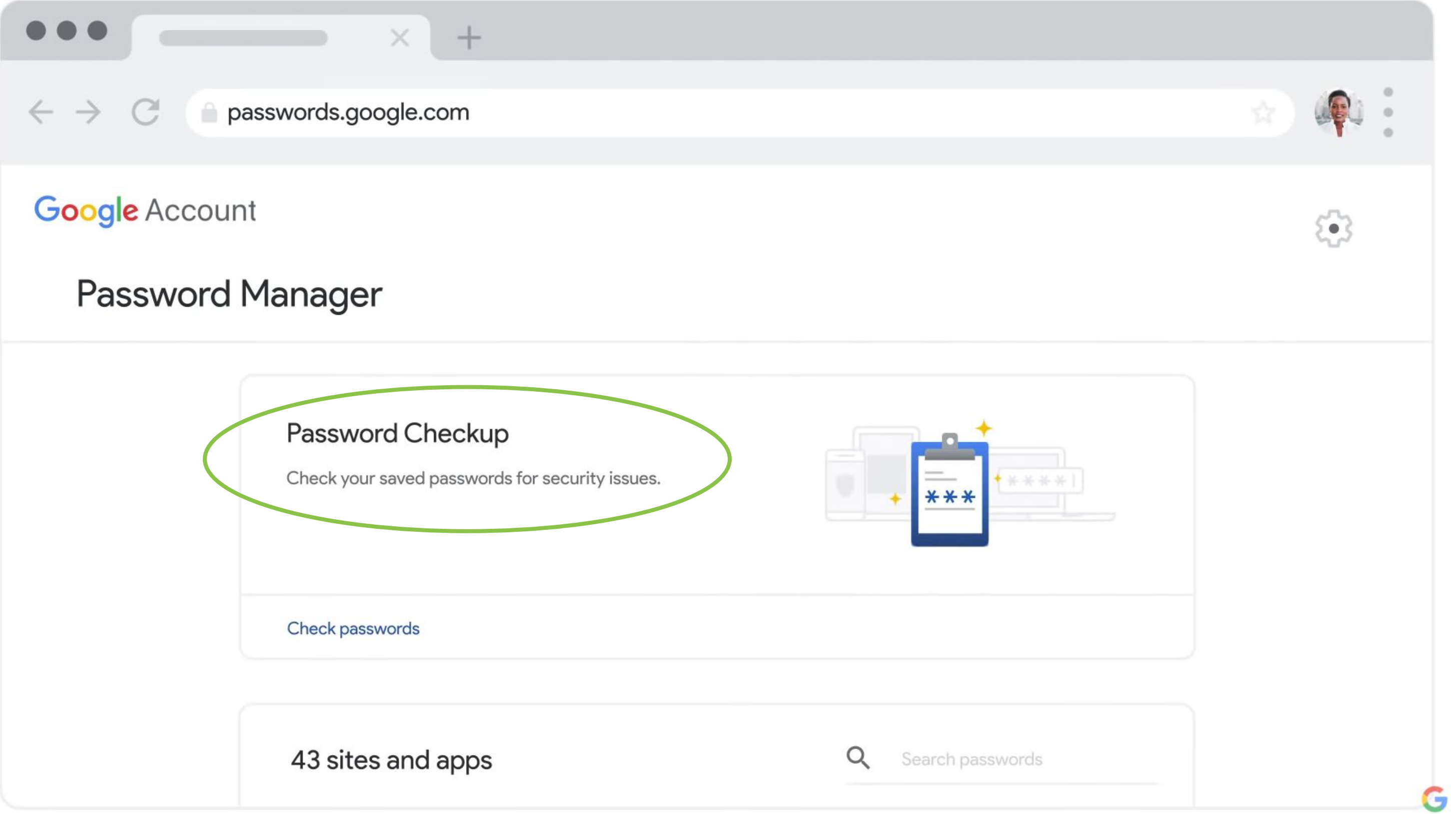
shoppingplatform.com

movieplatform.com

workoutplatform.com

passwords - password manager

Check Passwords



passwords.google.com

Google Account

Password Manager

Password Checkup

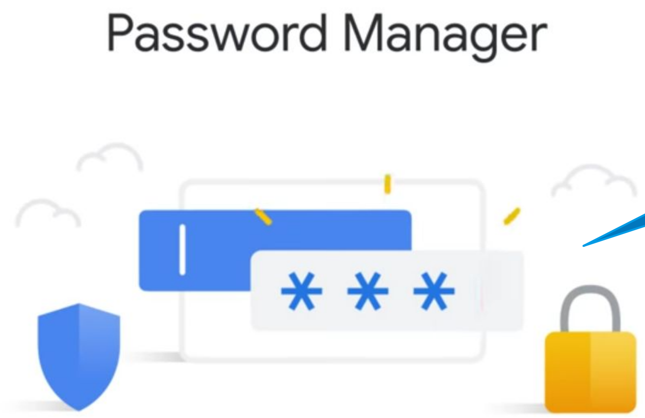
Check your saved passwords for security issues.

Check passwords

43 sites and apps

Search passwords

Protect Passwords



Google account password.
one strong password, to
protect all your other passwords.

Platform Integration

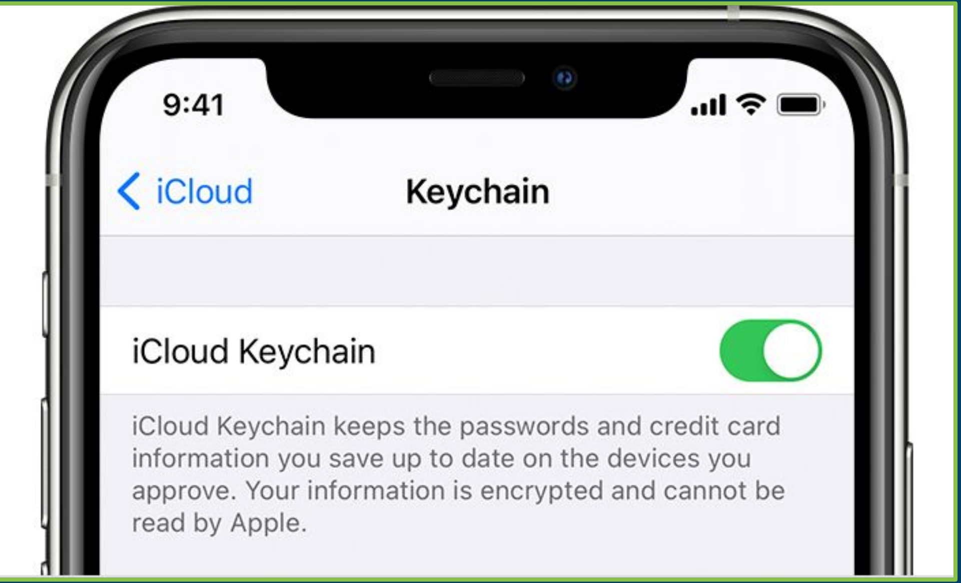
chrome

android

passwords - password managers

Others

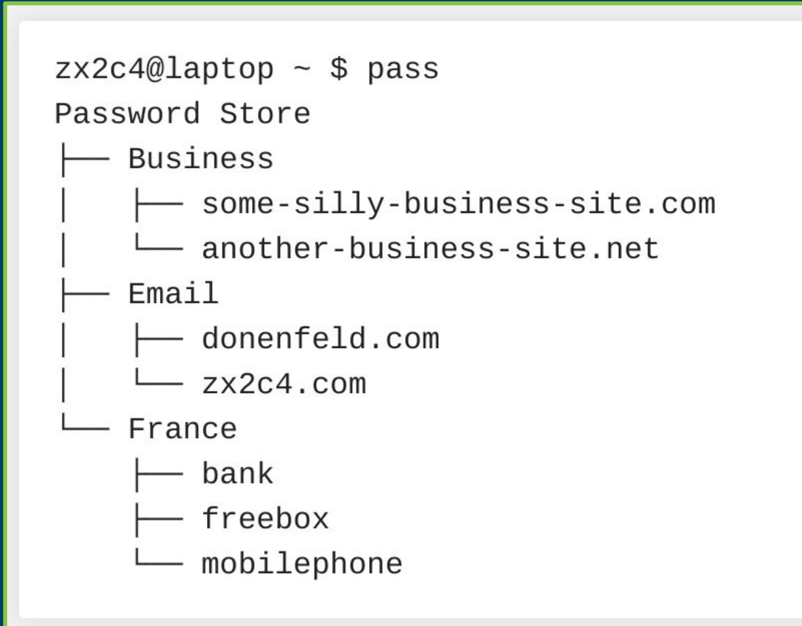
proprietary (Apple)



iCloud Keychain

iCloud Keychain keeps the passwords and credit card information you save up to date on the devices you approve. Your information is encrypted and cannot be read by Apple.

free (a little more in lecture 8)



```
zx2c4@laptop ~ $ pass
Password Store
├── Business
│   ├── some-silly-business-site.com
│   └── another-business-site.net
├── Email
│   ├── donenfeld.com
│   └── zx2c4.com
└── France
    ├── bank
    ├── freebox
    └── mobilephone
```

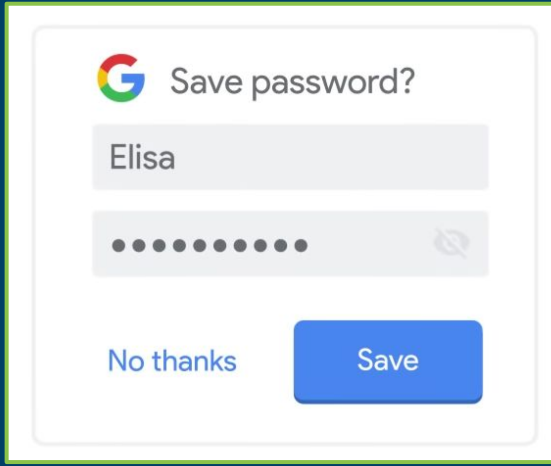
Summary

use a password manager!

- better than recycling passwords,
- better than using passwords generated by a predictable scheme.

which one: tradeoff

- trust (Google? Apple?)
- convenience (integration, already set up)
- your tech know-how (can you protect your password store?)



Save password?

Elisa

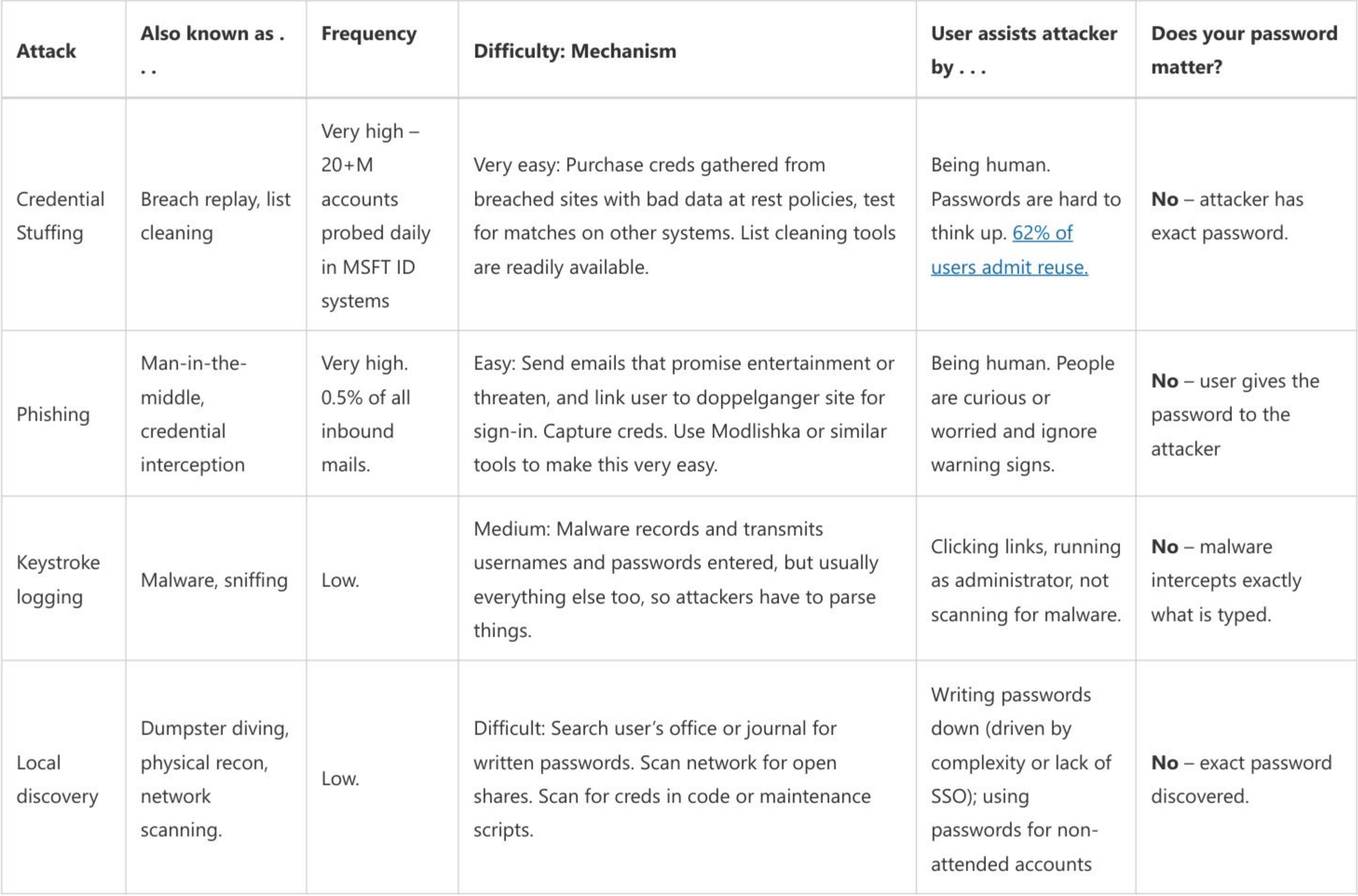
●●●●●●●●●●

MFA

multi-factor authentication

multi-factor authentication

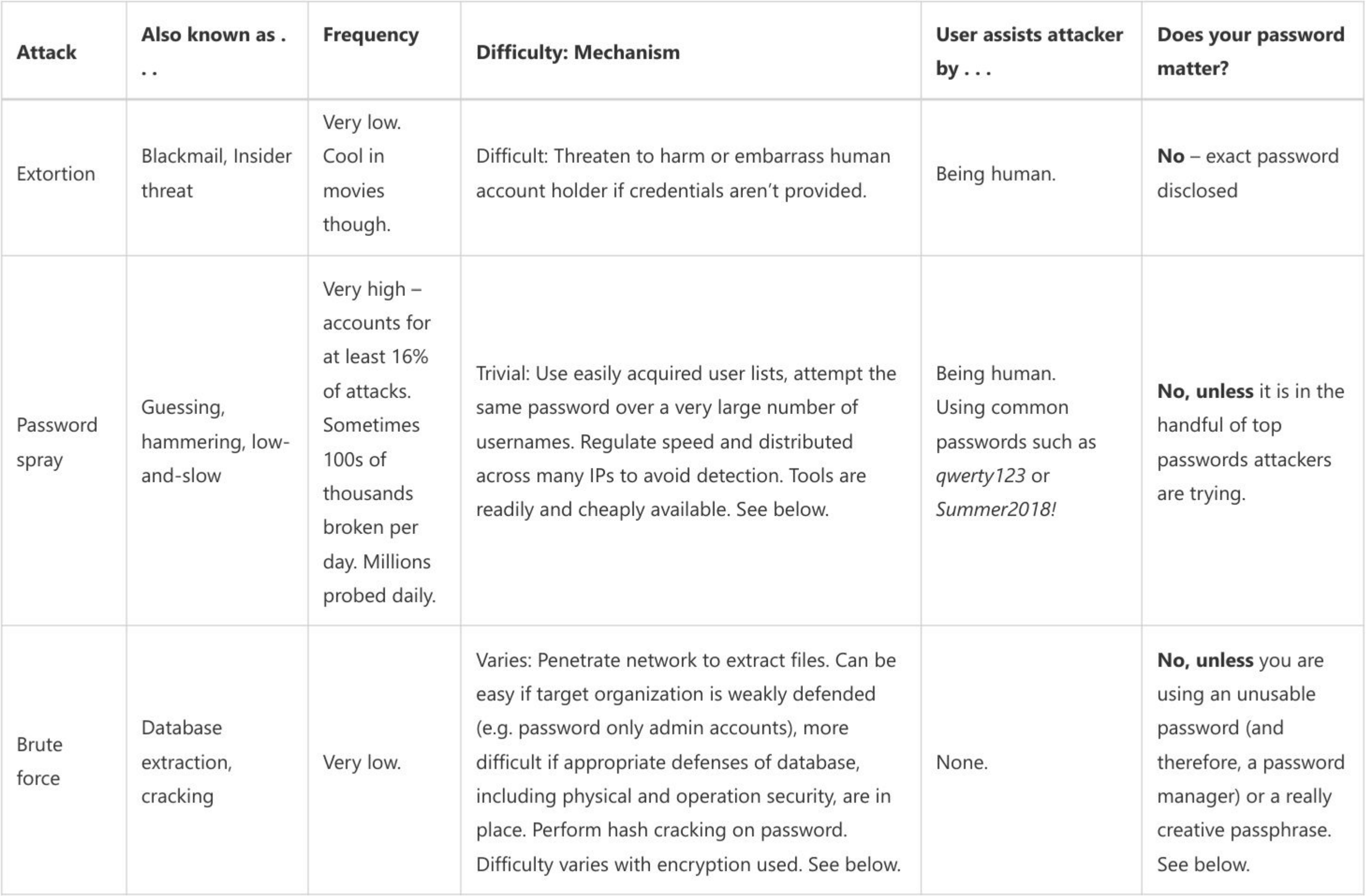
Passwords (alone) are insufficient



Attack	Also known as . . .	Frequency	Difficulty: Mechanism	User assists attacker by . . .	Does your password matter?
Credential Stuffing	Breach replay, list cleaning	Very high – 20+M accounts probed daily in MSFT ID systems	Very easy: Purchase creds gathered from breached sites with bad data at rest policies, test for matches on other systems. List cleaning tools are readily available.	Being human. Passwords are hard to think up. 62% of users admit reuse.	**No** – attacker has exact password.
Phishing	Man-in-the-middle, credential interception	Very high. 0.5% of all inbound mails.	Easy: Send emails that promise entertainment or threaten, and link user to doppelganger site for sign-in. Capture creds. Use Modlishka or similar tools to make this very easy.	Being human. People are curious or worried and ignore warning signs.	**No** – user gives the password to the attacker
Keystroke logging	Malware, sniffing	Low.	Medium: Malware records and transmits usernames and passwords entered, but usually everything else too, so attackers have to parse things.	Clicking links, running as administrator, not scanning for malware.	**No** – malware intercepts exactly what is typed.
Local discovery	Dumpster diving, physical recon, network scanning.	Low.	Difficult: Search user's office or journal for written passwords. Scan network for open shares. Scan for creds in code or maintenance scripts.	Writing passwords down (driven by complexity or lack of SSO); using passwords for non-attended accounts	**No** – exact password discovered.

multi-factor authentication

Passwords (alone) are insufficient



Attack	Also known as . . .	Frequency	Difficulty: Mechanism	User assists attacker by . . .	Does your password matter?
Extortion	Blackmail, Insider threat	Very low. Cool in movies though.	Difficult: Threaten to harm or embarrass human account holder if credentials aren't provided.	Being human.	**No** – exact password disclosed
Password spray	Guessing, hammering, low-and-slow	Very high – accounts for at least 16% of attacks. Sometimes 100s of thousands broken per day. Millions probed daily.	Trivial: Use easily acquired user lists, attempt the same password over a very large number of usernames. Regulate speed and distributed across many IPs to avoid detection. Tools are readily and cheaply available. See below.	Being human. Using common passwords such as *qwerty123* or *Summer2018!*	**No, unless** it is in the handful of top passwords attackers are trying.
Brute force	Database extraction, cracking	Very low.	Varies: Penetrate network to extract files. Can be easy if target organization is weakly defended (e.g. password only admin accounts), more difficult if appropriate defenses of database, including physical and operation security, are in place. Perform hash cracking on password. Difficulty varies with encryption used. See below.	None.	**No, unless** you are using an unusable password (and therefore, a password manager) or a really creative passphrase. See below.

multi-factor authentication

Defense in Depth - No Single Point of Failure

harder for attacker to obtain multiple factors

- A might crack your password
- B might steal your phone
- unlikely that A and B are the same person

drawbacks

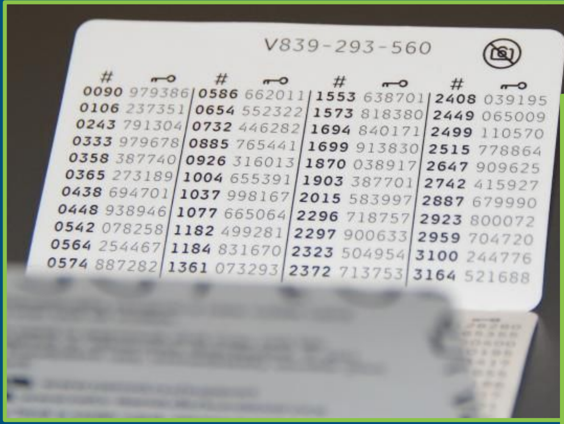
- more demanding for service provider
- more demanding for service consumer
  (extra steps, more SW to install, physical token to protect, etc.)

(historically,)
additional factor,
something you <u>have</u>:

(one-time password) tokens

we look at two types: based on

- challenge-response, or
- synchronized clocks



V839-293-560

#	🔑	#	🔑	#	🔑	#	🔑
0090	979386	0586	662011	1553	638701	2408	039195
0106	237351	0654	552322	1573	818380	2449	065009
0243	791304	0732	446282	1694	840171	2499	110570
0333	979678	0885	765441	1699	913830	2515	778864
0358	387740	0926	316013	1870	038917	2647	909625
0365	273189	1004	655391	1903	387701	2742	415927
0438	694701	1037	998167	2015	583997	2887	679990
0448	938946	1077	665064	2296	718757	2923	800072
0542	078258	1182	499281	2297	900633	2959	704720
0564	254467	1184	831670	2323	504954	3100	244776
0574	887282	1361	073293	2372	713753	3164	521688

(one-time password) tokens

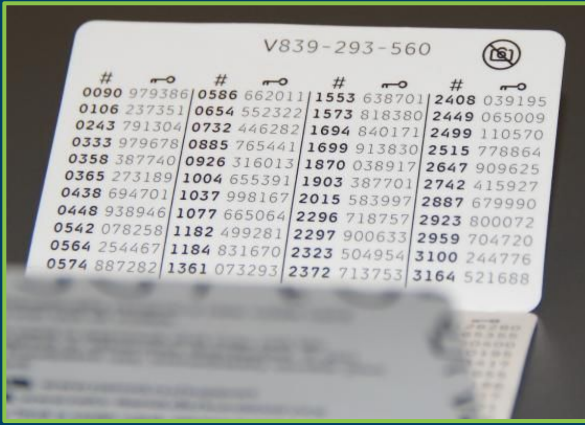
Challenge-Response, on paper (TAN)

Transaction Authentication Number

bank issues e.g. 50 TANs to you, on paper (pickup at bank; authenticate etc.).
to authorize a bank transaction, provide one TAN. once provided,
it is (marked by bank as) “spent” (one-time password). how **MFA:**

- TANs useless w/o login credentials,
- login credentials do not enable transfers w/o TANs

example: NemID nøglekort (key card)



V839-293-560

#	🔑	#	🔑	#	🔑	#	🔑
0090	979386	**0586**	662011	**1553**	638701	**2408**	039195
0106	237351	**0654**	552322	**1573**	818380	**2449**	065009
0243	791304	**0732**	446282	**1694**	840171	**2499**	110570
0333	979678	**0885**	765441	**1699**	913830	**2515**	778864
0358	387740	**0926**	316013	**1870**	038917	**2647**	909625
0365	273189	**1004**	655391	**1903**	387701	**2742**	415927
0438	694701	**1037**	998167	**2015**	583997	**2887**	679990
0448	938946	**1077**	665064	**2296**	718757	**2923**	800072
0542	078258	**1182**	499281	**2297**	900633	**2959**	704720
0564	254467	**1184**	831670	**2323**	504954	**3100**	244776
0574	887282	**1361**	073293	**2372**	713753	**3164**	521688

- indexed (iTAN); bank asks for a *specific* TAN.

problems:

(one-time password) tokens

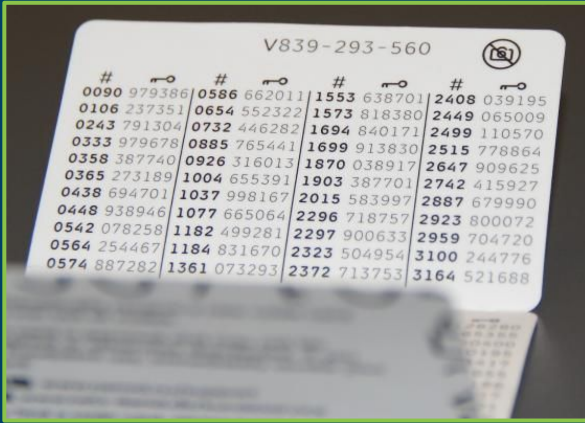
Transaction Authentication Number

Challenge-Response, on paper (TAN)

bank issues e.g. 50 TANs to you, on paper (pickup at bank; authenticate etc.).
to authorize a bank transaction, provide one TAN. once provided,
it is (marked by bank as) “spent” (one-time password). how **MFA:**

- TANs useless w/o login credentials,
- login credentials do not enable transfers w/o TANs

example: NemID nøglekort (key card)



V839-293-560

#	🔑	#	🔑	#	🔑	#	🔑
0090	979386	0586	662011	1553	638701	2408	039195
0106	237351	0654	552322	1573	818380	2449	065009
0243	791304	0732	446282	1694	840171	2499	110570
0333	979678	0885	765441	1699	913830	2515	778864
0358	387740	0926	316013	1870	038917	2647	909625
0365	273189	1004	655391	1903	387701	2742	415927
0438	694701	1037	998167	2015	583997	2887	679990
0448	938946	1077	665064	2296	718757	2923	800072
0542	078258	1182	499281	2297	900633	2959	704720
0564	254467	1184	831670	2323	504954	3100	244776
0574	887282	1361	073293	2372	713753	3164	521688

- indexed (iTAN); bank asks for a *specific* TAN.

problems: copy the TANs, replay attack, man-in-the-middle attack, ...

(one-time password) tokens

Challenge-Response, hardware token

bank issues a hardware token you (pickup at bank; authenticate etc.).
to authorize a bank transaction, unlock token & input challenge from bank.
once provided, token generates response (which you send to bank).

example: Länsförsäkringar säkerhetsdosa

similar to iTAN.

- harder to read (unlock),
- harder to copy/steal w/o it being noticed.

(one-time password) tokens

Synchronized Clocks (hardware token)

to authorize a bank transaction, press button on token.
token generates a number (which you send to bank).

how it works: bank generates secret. embeds it in token.
token has a clock, synchronized w/ bank's clock.
token & bank both generate hash of secret + current time.

example: RSA SecurID

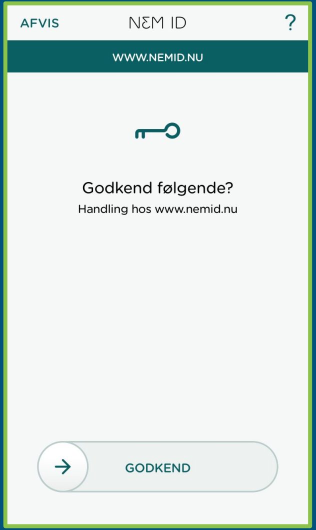
example: NemID nøgleviser

protected against replay attacks (OTPs expire).

(modern)
additional factor,
something you have:

we look at two types:
- apps, or
- USB key

(one-time password) tokens


AFVIS

NEM ID

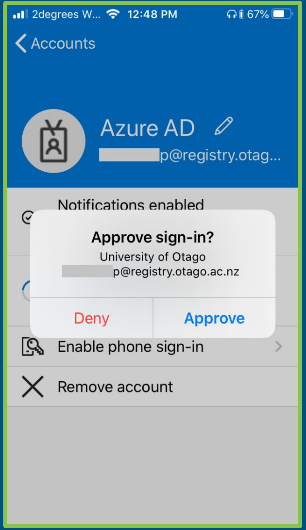
?

WWW.NEMID.NU

Godkend følgende?

Handling hos www.nemid.nu

GODKEND


2degrees W... 12:48 PM 67%

Accounts

Azure AD

[redacted]p@registry.otag...

Notifications enabled

Approve sign-in?

University of Otago

[redacted]p@registry.otago.ac.nz

Deny | Approve

Enable phone sign-in

Remove account

(one-time password) tokens

App

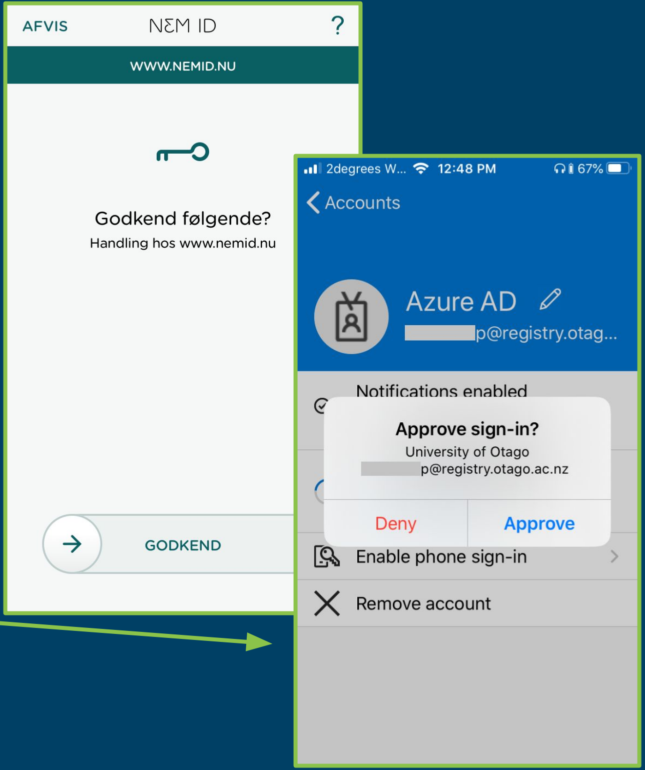
app receives a request for authorization.
you grant it by swiping in the app.

how it works: implemented using either, or both, of

- challenge-response (cryptographic keys)
- synchronized clocks

example: NemID nøgleapp
example: Microsoft Authenticator

problems:



Godkend følgende?
Handling hos www.nemid.nu

[redacted]p@registry.otag...

Notifications enabled

Approve sign-in?
University of Otago
[redacted]p@registry.otago.ac.nz

(one-time password) tokens

App

app receives a request for authorization.
you grant it by swiping in the app.

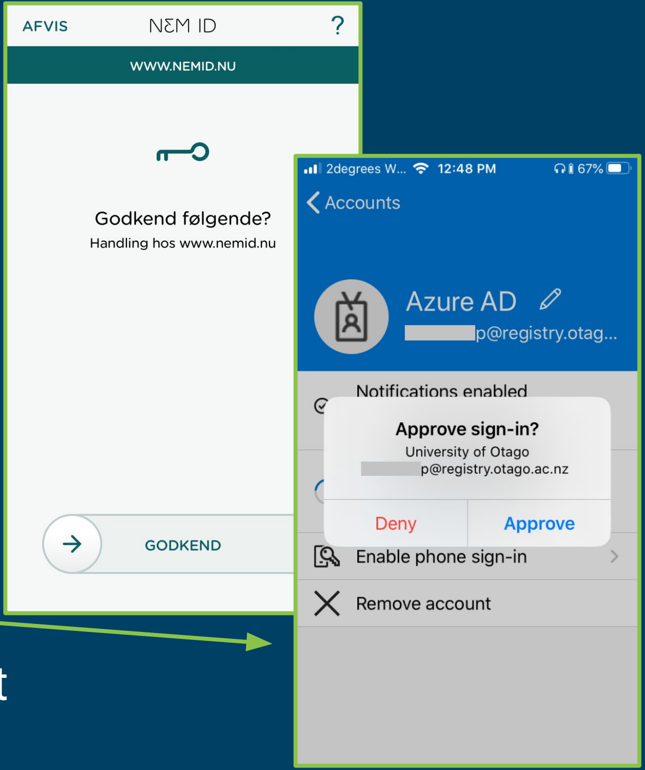
how it works: implemented using either, or both, of

- challenge-response (cryptographic keys)
- synchronized clocks

example: NemID nøgleapp
example: Microsoft Authenticator

problems: still vulnerable to man-in-the-middle, + rootkit

next slide



AFVIS NEM ID ?

WWW.NEMID.NU

Godkend følgende?
Handling hos www.nemid.nu

GODKEND

2degrees W... 12:48 PM 67%

Accounts

Azure AD
[redacted]p@registry.otag...

Notifications enabled

Approve sign-in?
University of Otago
[redacted]p@registry.otago.ac.nz

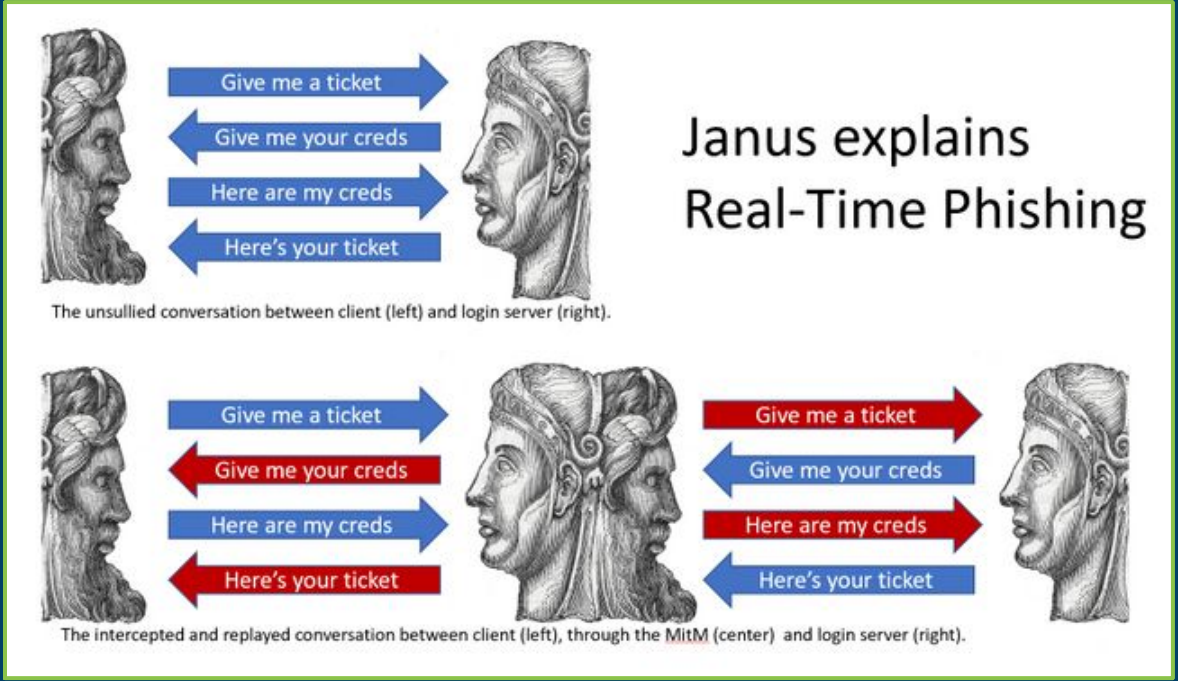
Deny Approve

Enable phone sign-in

Remove account

(one-time password) tokens

App, man in the middle phishing attack



The unsullied conversation between client (left) and login server (right).

The intercepted and replayed conversation between client (left), through the MitM (center) and login server (right).

(one-time password) tokens

USB key

service (e.g. Web client, in your browser) requests your authorization.
you grant it by touching a USB key.

how it works: implemented using either, or both, of

- challenge-response (cryptographic keys)
- synchronized clocks

example: YubiKey

harder to hack (separate device). **thwarts man-in-the-middle** (USB key is bound to origin at account creation)

(one-time password) tokens

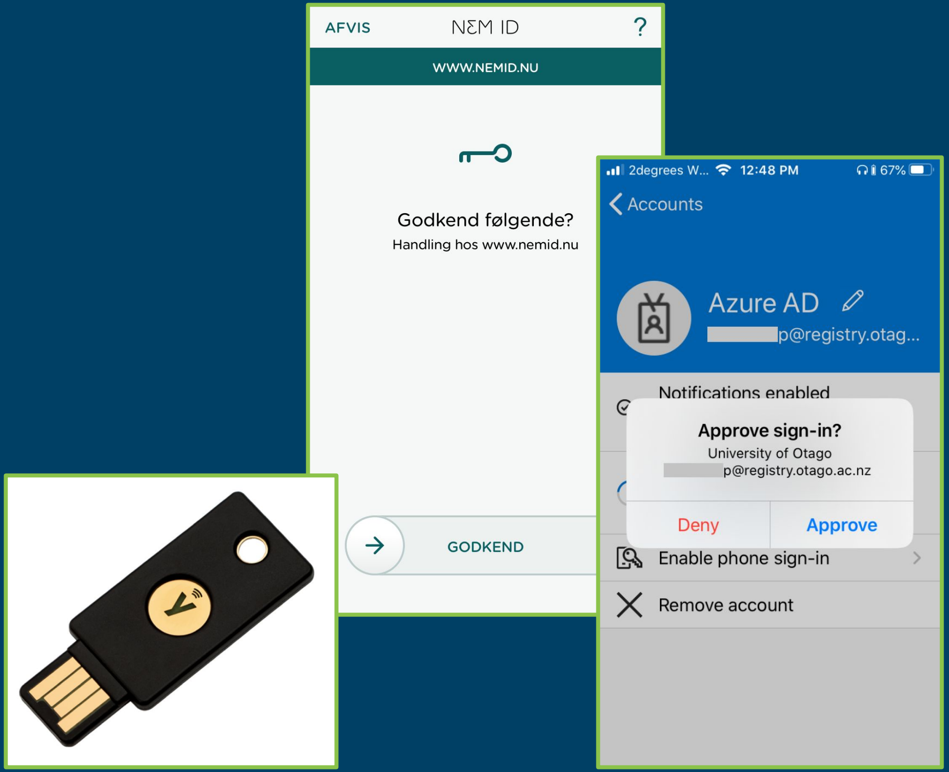
Summary

use multi-factor authentication!

- much harder to hack an account.

at least: authenticator app
ideally: USB-key (phishing protection).

adoption for both is picking up fast.
becoming standard / **expected**.



AFVIS NΣM ID ?

WWW.NEMID.NU

Godkend følgende?
Handling hos www.nemid.nu

GODKEND

2degrees W... 12:48 PM 67%

Accounts

Azure AD
[redacted]p@registry.otag...

Notifications enabled

Approve sign-in?
University of Otago
[redacted]p@registry.otago.ac.nz

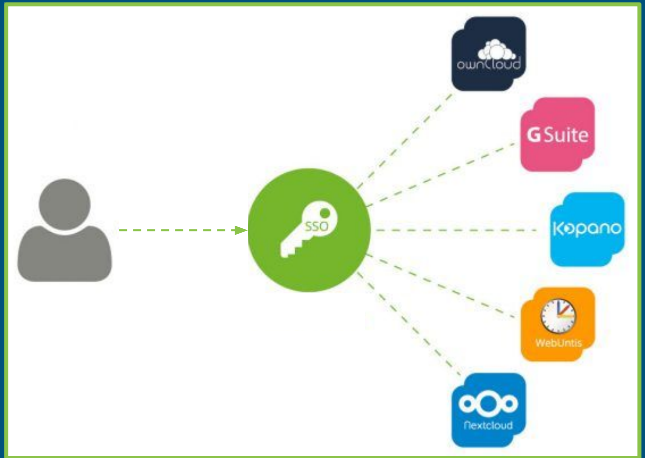
Deny Approve

Enable phone sign-in

Remove account

SSO

Single Sign-On



Single Sign-On

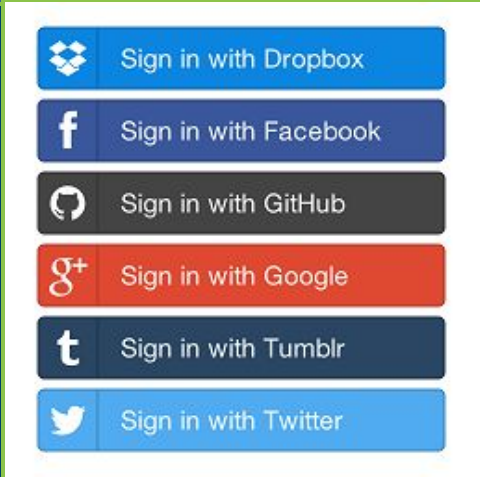
Minimum Exposure - Reduce Attack Surface

having many accounts (with many services) is.

- **demanding:** set up MFA for each of them.
- **not secure:** more accounts ⇒ larger attack surface.

instead, level of indirection: **identity as a service** (IDaaS).

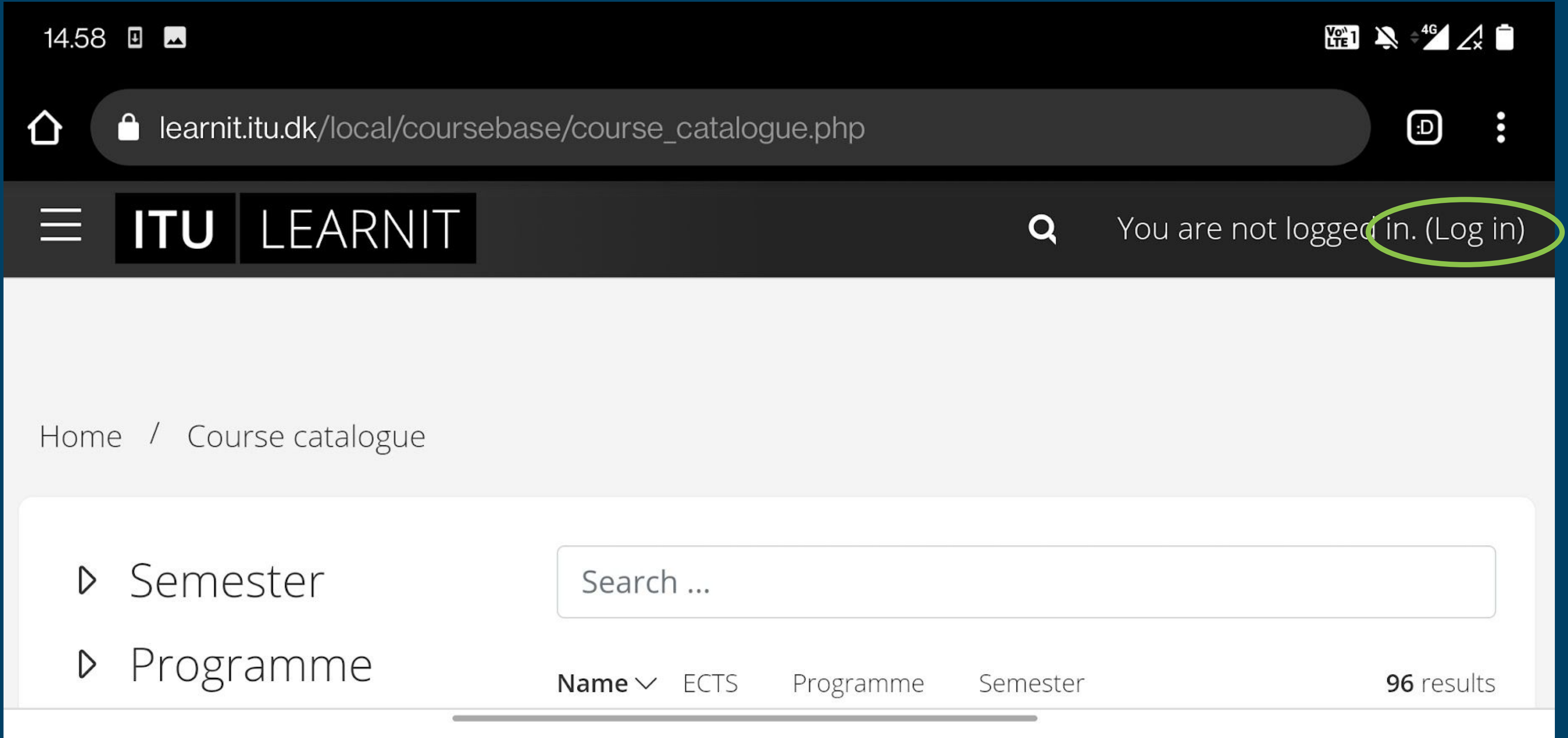
- less demanding
- smaller attack surface
- **risk:** single point of failure: the **identity provider**
  (i.e. if they're taken down, get hacked, etc.)
  (easier to make one thing bullet-proof than many.)



how does it work?

Single Sign-On

SSO at ITU, LearnIT



14.58

learnit.itu.dk/local/coursebase/course_catalogue.php

ITU LEARNIT

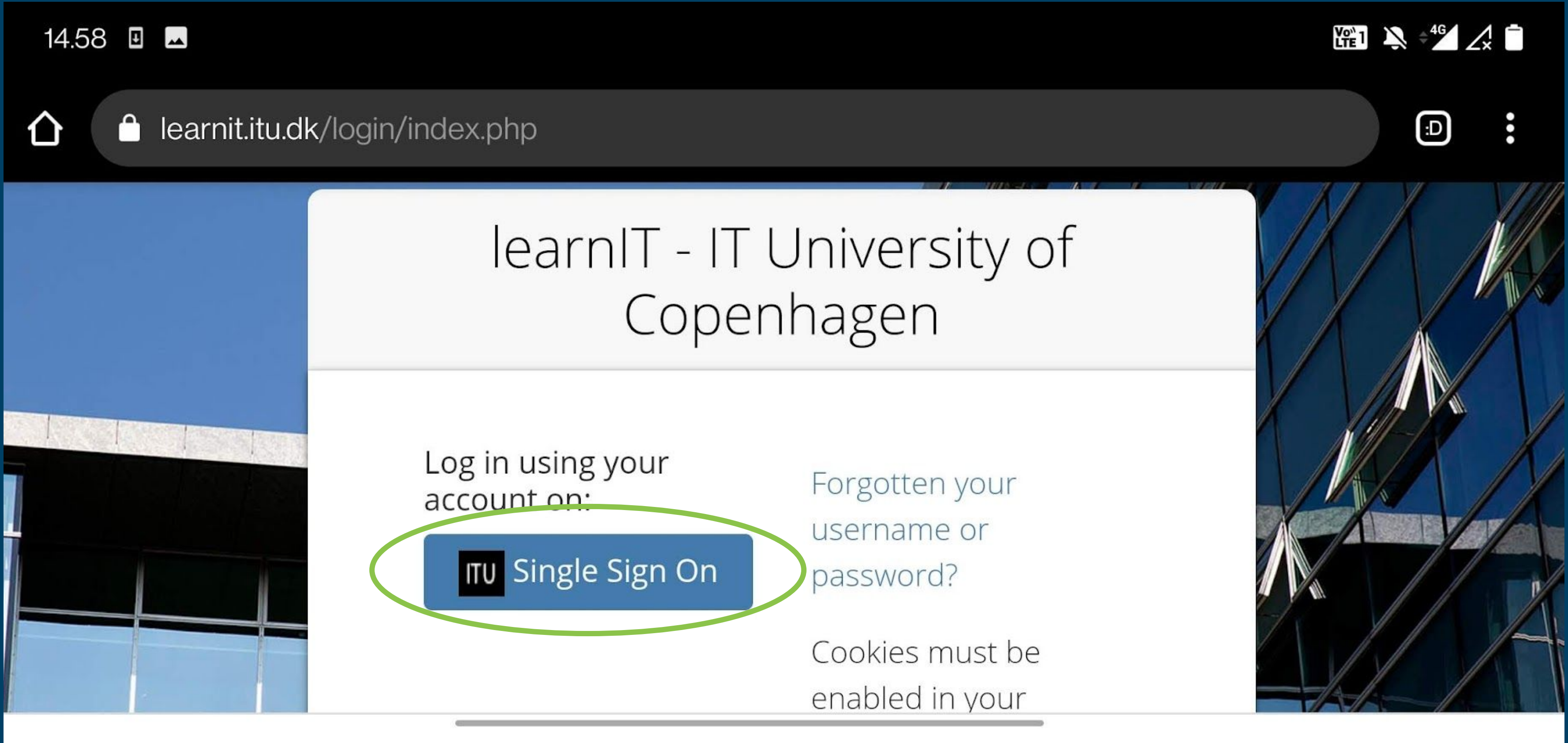
You are not logged in. (Log in)

Home / Course catalogue

Name	ECTS	Programme	Semester	96 results

Single Sign-On

SSO at ITU, LearnIT: log in “at the app”



14.58

learnit.itu.dk/login/index.php

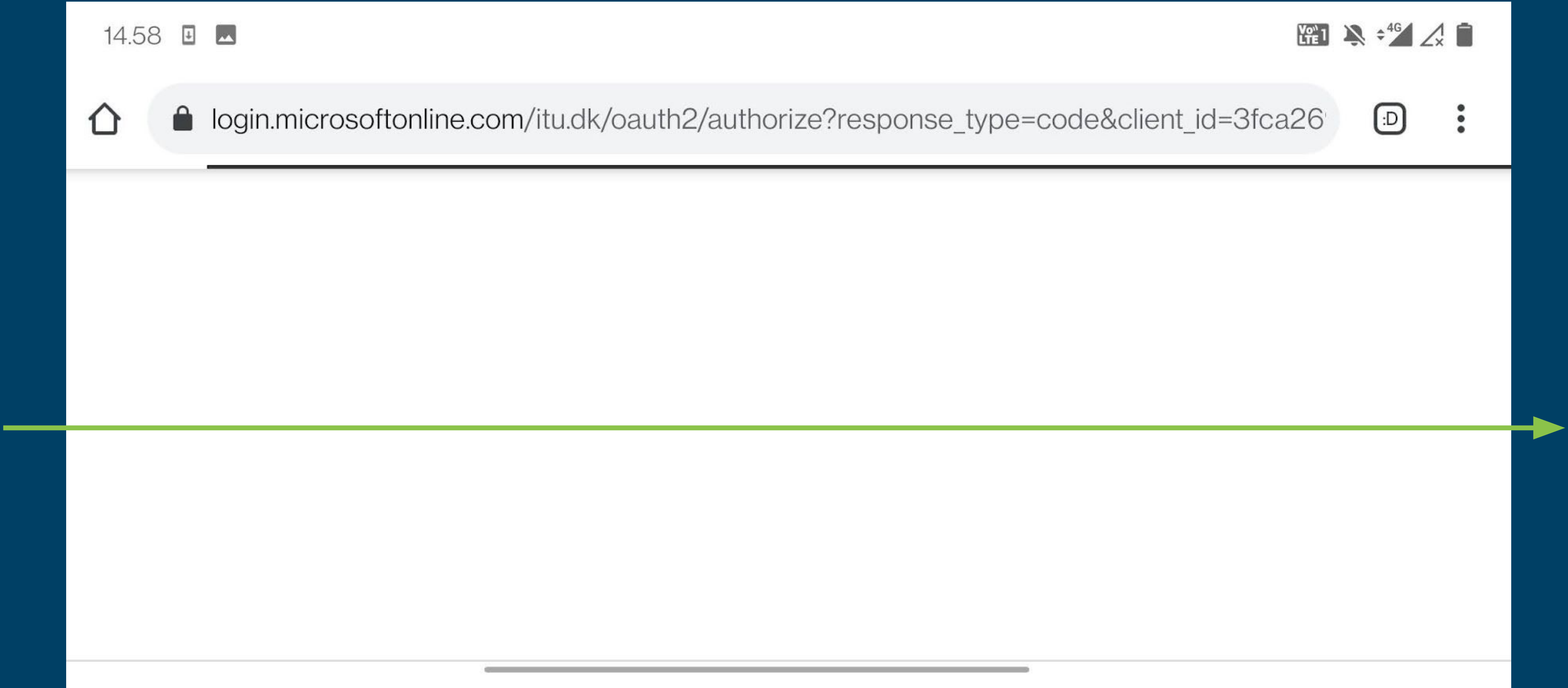
learnIT - IT University of Copenhagen

Log in using your account on:

Forgotten your username or password?

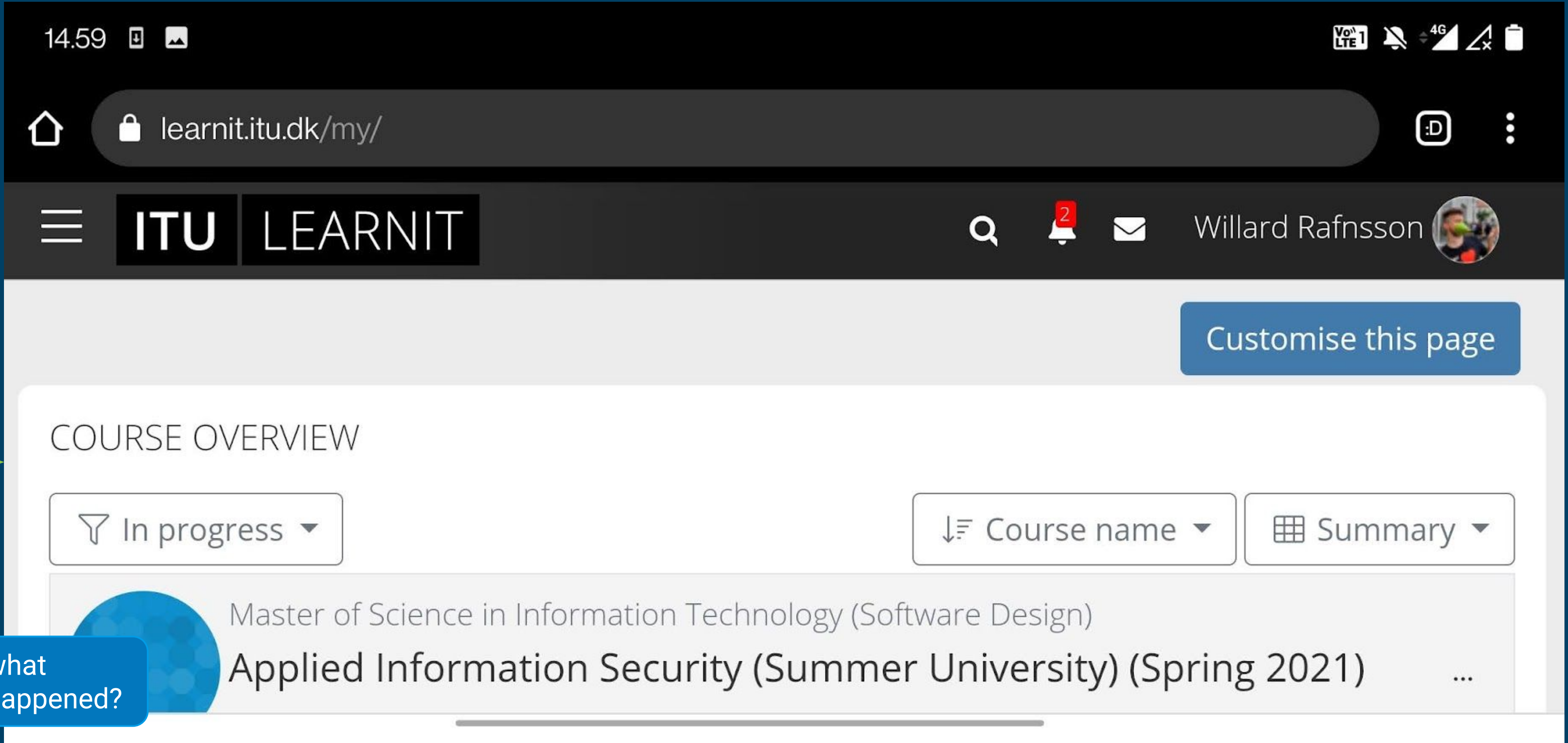
Cookies must be enabled in your

SSO at ITU, LearnIT: request forwarded...



login.microsoftonline.com/itu.dk/oauth2/authorize?response_type=code&client_id=3fca26

SSO at ITU, LearnIT: … and you're logged in.

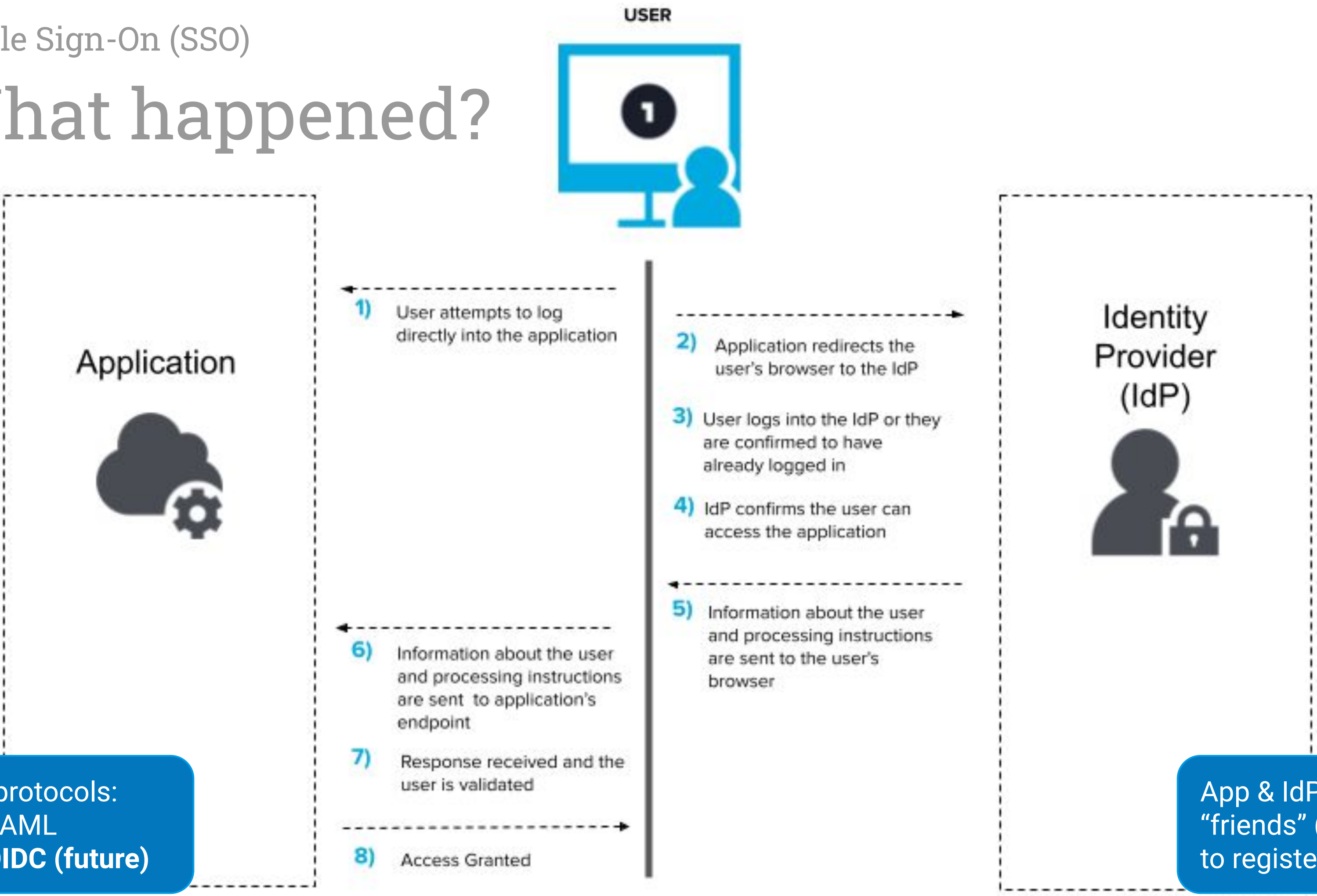


14.59
learnit.itu.dk/my/
ITU LEARNIT
2
Willard Rafnsson
Customise this page
COURSE OVERVIEW
In progress
Course name
Summary
Master of Science in Information Technology (Software Design)
Applied Information Security (Summer University) (Spring 2021)
...

what happened?

Single Sign-On (SSO)

What happened?



USER

Application

Identity Provider (IdP)

1) User attempts to log directly into the application

2) Application redirects the user's browser to the IdP

3) User logs into the IdP or they are confirmed to have already logged in

4) IdP confirms the user can access the application

5) Information about the user and processing instructions are sent to the user's browser

6) Information about the user and processing instructions are sent to application's endpoint

7) Response received and the user is validated

8) Access Granted

today's protocols:

- SAML
- **OIDC (future)**

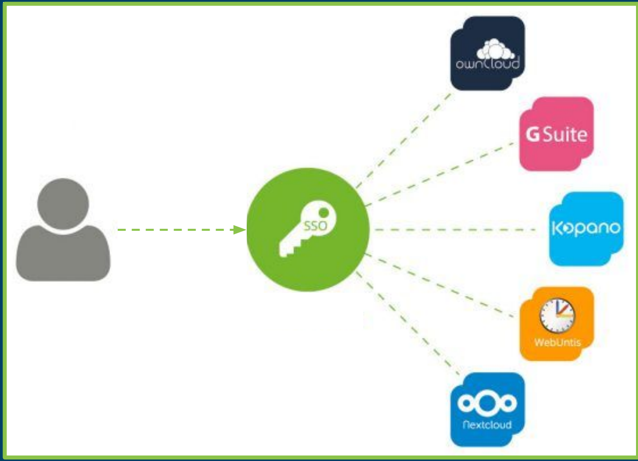
App & IdP need to be "friends" (App needs to register at IdP)

Single Sign-On

Summary

use SSO!

- more secure (smaller attack surface)
- less demanding

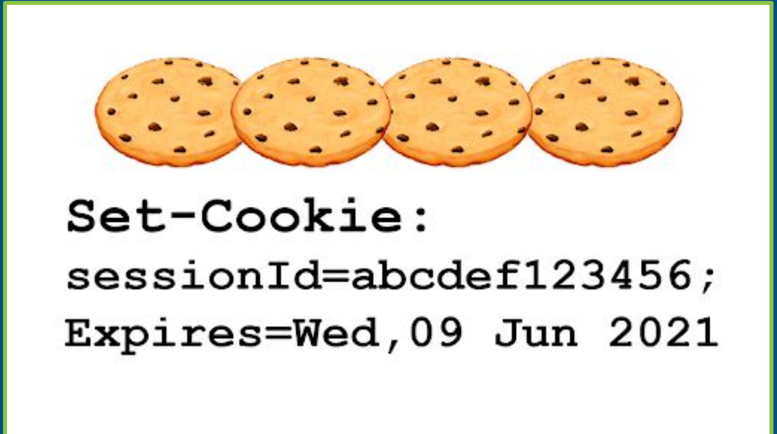


developer: *don’t roll your own authentication.*
(bullet-proofing authentication is hard)

user: only use an identity provider that you **trust**.

login

Tokens (Cookies)



```
Set-Cookie:
sessionId=abcdef123456;
Expires=Wed,09 Jun 2021
```

Authentication on the Web

know / have / are

recall: inconvenient to do enrollment-level validation (physical presence, etc.) when you **login**. instead, check fewer, **easy to protect & hard to spoof** attributes.

web: server won't know that two requests from same host are from <u>you</u>. you need to authenticate **per request**. for **know/are**, that's inconvenient.

solution: **tokens**. exchange **know** / **are** for a token (**have**, in SW).

- future requests: authenticate by including token in request.
- **assumption:** token can only be obtained from the server (by)

today: specific kind of token (cookie). more general ones later (bearer token)

Tokens (Cookies)

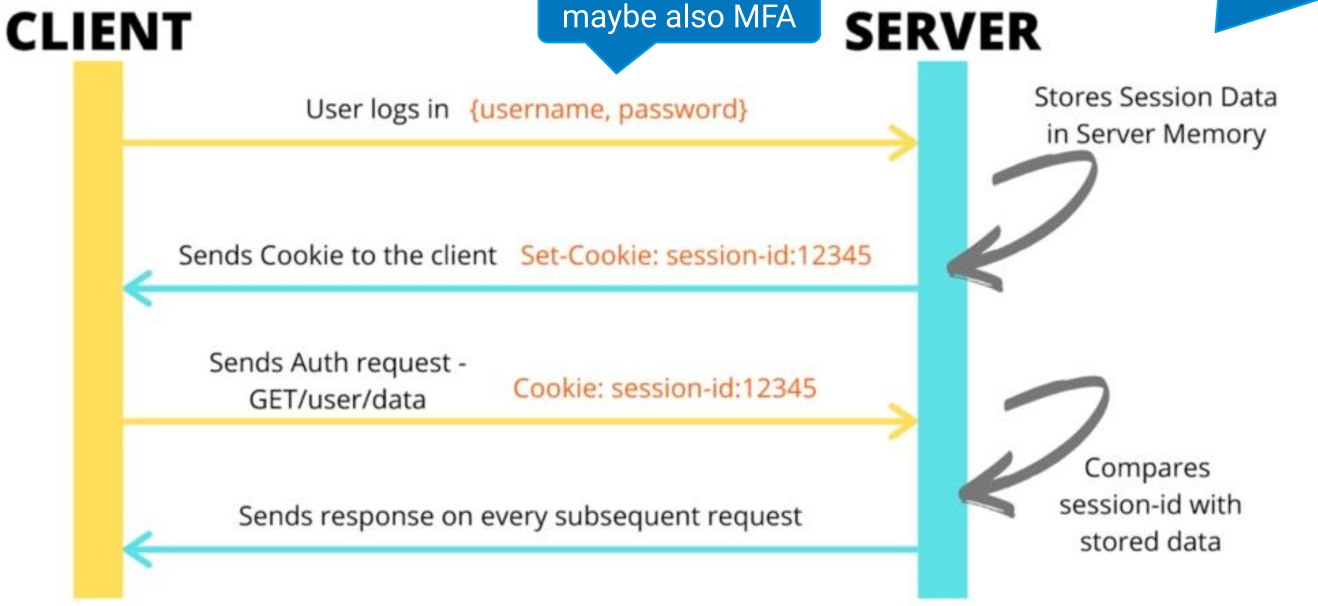
Cookie-Based Authentication

stage in checkout procedure, shopping cart, ...

need to store information about the session? (Session-Based Authentication)

only need to know who this is? no need to store stuff.

maybe also MFA



CLIENT

SERVER

User logs in {username, password}

Stores Session Data in Server Memory

Sends Cookie to the client Set-Cookie: session-id:12345

Sends Auth request - GET/user/data

Cookie: session-id:12345

Compares session-id with stored data

Sends response on every subsequent request

or Encrypt session-id

or Decrypt session-id

Tokens (Cookies)

Cookie Security Tips

how to secure cookie-based authentication:

- **use the `HttpOnly` attribute**
  prevents JavaScript from accessing it in the client
- **use a short lifetime (with `Expires=`)**
  limits impact of a stolen cookie
- **set `SameSite=Strict` or `Lax`**
  prevents cookie from being shipped to third-parties on cross-site requests.

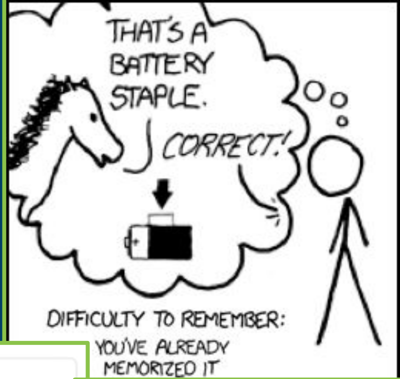
Summary

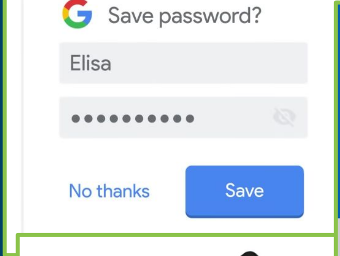
user:

- use a password manager (w/ a strong password)
- use multi-factor authentication
- use single sign-on

developer:

- good password policy
- provide multi-factor authentication
- don’t roll your own authentication
- secure that cookie

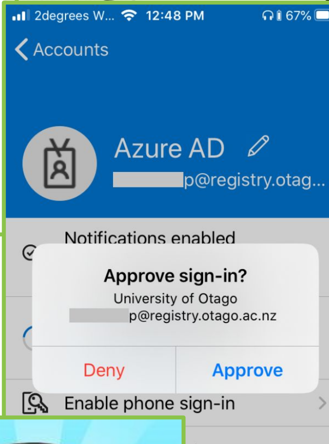



Save password?

Elisa

••••••••••

No thanks Save


2degrees W... 12:48 PM 67%

Accounts

Azure AD

[redacted]p@registry.otag...

Notifications enabled

Approve sign-in?

University of Otago

[redacted]p@registry.otago.ac.nz

Deny Approve

Enable phone sign-in