Authentication

Introduction to Security (192.019)

Mauro **Tempesta**

Security & **Privacy** Research Unit (192-06)
https://**secpriv.wien**

Identification

Identification is the process of stating the identity of an individual by providing distinguishing information for this person

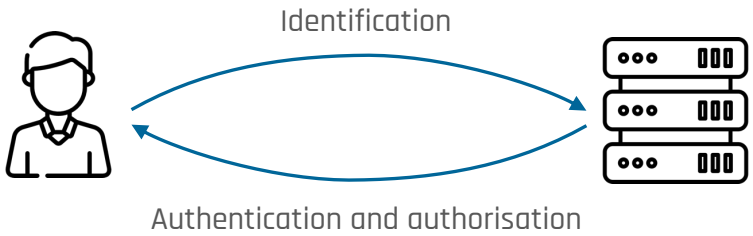
- **Physical world**: indication of name, surname, place and date of birth
  - **Uniqueness** of the identifying information guaranteed by registry offices
- **Digital world**: IT systems determine how users can be uniquely identified
  - E-mail address
  - Choice of a username by each user
  - Automatic assignment of a name to each user

Authentication

Authentication is the act of verifying the identity of a person

- **Physical world**: presentation of an identity card
  - Provided information is compared with that in the document
- **Digital world**:
  - Usage of passwords
  - Biometrics (face recognition, fingerprint)
  - Certificate verification
  - ...

Steps to Access an IT System



1. **Identification**: the user provides information about the identity and an evidence of the authenticity of such information
2. **Authentication**: the system verifies the correctness of the provided identifying data
3. **Authorisation**: access to the system is provided after successful authentication

Classes of Authentication Methods

Knowledge

- Verification of an information known only to the user
- Examples: passwords, PINs, security questions

Ownership

- Verification of the possession of an object owned by the user
- Examples: smartcards, TAN lists, hardware tokens, SIM card, certificates, authenticator apps

Inherence

- Verification of biometric characteristics of the user
- Examples: fingerprint, face recognition, iris recognition, DNA

Overview of Authentication Methods

- **Passwords**
  - The simplest and most common authentication method
  - The password must be **communicated securely** (e.g., over TLS) to prevent leakages
  - Rules enforcing password quality should be enforced

- **One-time passwords**
  - Every password can be used only **once**
  - Passwords are either **precomputed** (e.g., TAN lists) or can be computed **on demand** by the user (e.g., hardware tokens)

Overview of Authentication Methods

- **Challenge-response methods**
  - Usually built on cryptography schemes
  - Each user knows a secret or possesses a secret key
  - Ability to perform **cryptographic operations** proves the knowledge of the required secret / possession of the required key

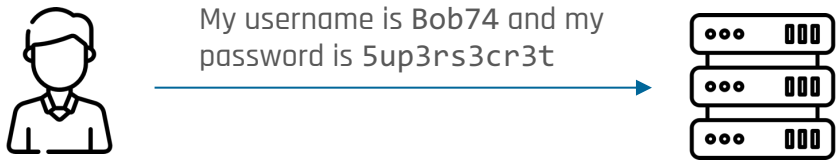
- **Biometry**
  - Both passive (e.g., fingerprint, face recognition) and active methods (e.g., voice recognition, typing behaviour)
  - Not suited for authentication over the Internet (**privacy** issues)
  - Usable for physical **access control** (e.g., doors)

Password-based Authentication

Password-based Authentication

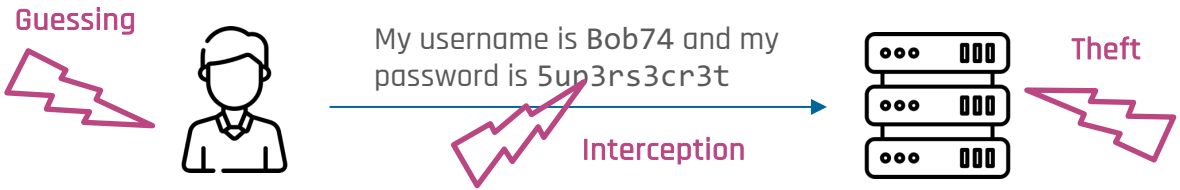
- Most **widespread** authentication scheme used on the Internet
  - User registers in advance to the system with username and password
  - Knowledge of the password is tested upon login
  - Requires the management of a **users' list** with corresponding passwords

- Security **crucially** depends on:
  - the **secrecy** and **quality** of the password
  - how the password is **stored** and **transmitted**



Attack Vectors

- **Guessing** of the correct password (social engineering, trial-and-error)
- **Interception** during the input or the transmission
- **Theft** of the passwords' list stored on the system



Attacks against Passwords

- Attack types
  - Offline attacks: the attacker managed to obtain access to the passwords list and tries to recover the (typically protected) stored passwords
  - Online attacks: the attacker interacts directly with the service where they want to authenticate (e.g., via the login form on a website)

- Strategies for cracking passwords
  - Dictionary-based attacks: trying out commonly used passwords contained in a passwords list (the so-called dictionary)
    - Possibly combined with social engineering techniques to choose passwords based on the environment or the habits of the target user
  - Brute-force attacks: systematic testing of all possible passwords

Dictionary-based Attacks

- The attacker prepares / downloads a list of **commonly used passwords**
  - Typically constructed from credentials **leaked** from vulnerable online services
  - The list might be complemented with words **related to the user** (kids names, favourite soccer team, ...)

- The attacker tries to login using all passwords in the list
  - Intuition: many users choose **weak / predictable** passwords
  - **Variations** of the passwords in the list might be tested as well
    - Addition of numbers / punctuation
    - Replacement of letters with similar-looking numbers (e.g., S -> 5, E -> 3)

Brute-force Attacks

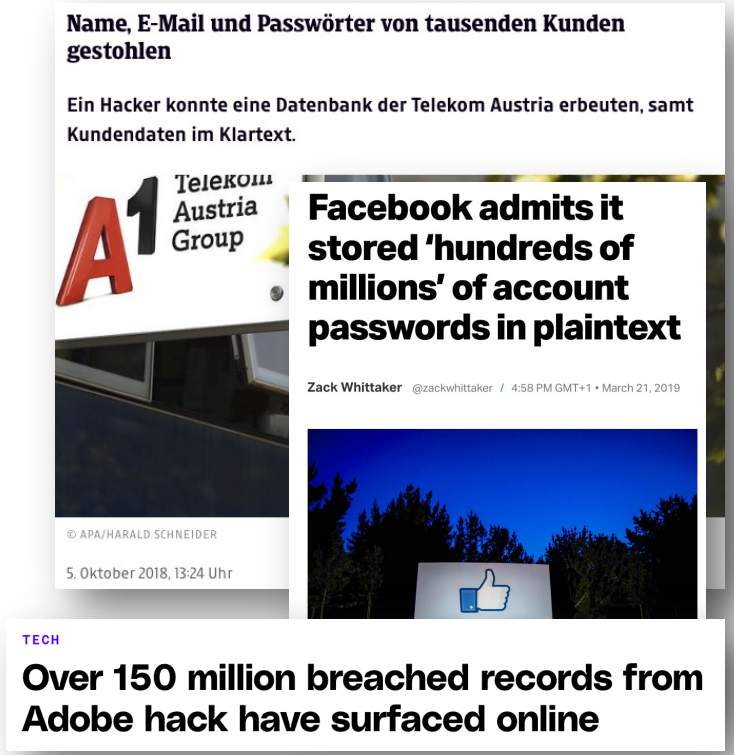
- Attacker generates and tests **all** possible passwords!
  - Theoretically always possible (with infinite time)!
  - In practice, only reasonable for **offline attacks**
  - Risk depends on the attacker's resources and the number of passwords

Assuming **one million** attempts per second!

Alphabet	Password length	# combinations	Full search time
0-9 (10 chars)	8	10^8 = 100.000.000	100 seconds
	10	10^{10} = 10.000.000.000	2,8 hours
A-Z a-z 0-9 (62 chars)	8	62^8 = 56.800.235.584	6,9 years
	10	62^{10} = 839.299.365.868.340.244	26.614 years
A-Z a-z 0-9 ()[]{} ?!$%&/=*+~,.;:<> -_ (86 chars)	8	86^8 = 2.992.179.271.065.856	94,88 years
	10	86^{10} = 22.130.157.888.803.070.976	701.743 years

Password Storage – How NOT to

- Passwords in cleartext
  - If the passwords list is leaked, the security of all users is compromised!
- Encrypted passwords
  - Passwords are stored in encrypted form
  - Decryption key must be available on the system to implement the login procedure
    - If the attacker leaks the key besides the list, all users' passwords can be recovered!
  - Adobe Leak (2013): passwords encrypted with 3DES (predecessor of AES) in ECB mode



Password Storage with Hash Functions

- The **hash** of the user's password is stored in the password list
  - Hash must be computed with a **cryptographic hash function**
  - **One-way property** ensures that the password cannot be easily recovered

- Implementation of the **login procedure**
  - User provides username and password
  - System computes the hash value of the password
  - Login successful if the stored hash **matches** the computed one

Security of Password Storage with Hash Functions

- Leakage of the passwords file **does not** necessarily compromise security
  - The attacker needs the **password** to authenticate, not its hash value

- Possible attack strategy
  - **Offline dictionary / brute-force attack**
    - Attacker computes the hash of common / all passwords and compares them with those hashes in the passwords file (hash computation is quick)
    - Can be automated by using tools such as *John the Ripper* or *hashcat*
  - **Usage of rainbow tables**
    - **Precomputed tables** mapping hash values to corresponding inputs
    - Usually contain only common words, short strings and commonly used passwords
    - Reuse is possible if the hash function does not change

Salt and Pepper

- **Salt** and **pepper** are randomly generated strings that are concatenated to the user's password before hashing
  - The salt is **different** for every user and stored in the **passwords list**
  - The pepper is **secret**, shared for all users and must be safely stored **outside** the passwords list (e.g., in a Hardware Security Module)
    - **Strong requirement**: many systems only use salting

$$\text{Hash} = h(\text{Password} \parallel \text{Salt} \parallel \text{Pepper})$$

User	Hash	Salt	Pepper
alice@gmx.de	07 A9 6E 3E 0D … BD	28 64 31 12 95 … EB	EF B4 85 02 1F … EA
bob@yahoo.com	E7 26 5E 61 D4 … FA	90 51 12 68 43 … 30	
mauro@gmail.com	78 A9 2B 0E E7 … FC	DB C6 DB A0 57 … 5C	

Advantages of Salt and Pepper

- Salting **complicates** the usage of **rainbow tables**
  - An attacker should precompute hashes of common passwords **for each possible salt**
  - Salt of n bits -> 2^n possible hashes for a given password (storage problems)
- Salting **hides** the phenomenon of **password reuse** by different users
  - Hashing of the same password with different salts leads to different hash values

- The usage of pepper makes even the cracking of a **single** hash **infeasible**
  - Even if the initial password is **very weak**!
  - NIST recommends a pepper size of at least 112 bits
  - **No security advantage** if the pepper is **public**

Hash Stretching

- Hash stretching makes the computation of the password hash more intensive by **repeating** the application of the hash function **multiple times**
  - Suggested **iteration count c** ranges between 100k and one million
  - Users notice a small delay during login (<1 second)
  - Computational time required for dictionary / brute-force attacks **increased by factor c**

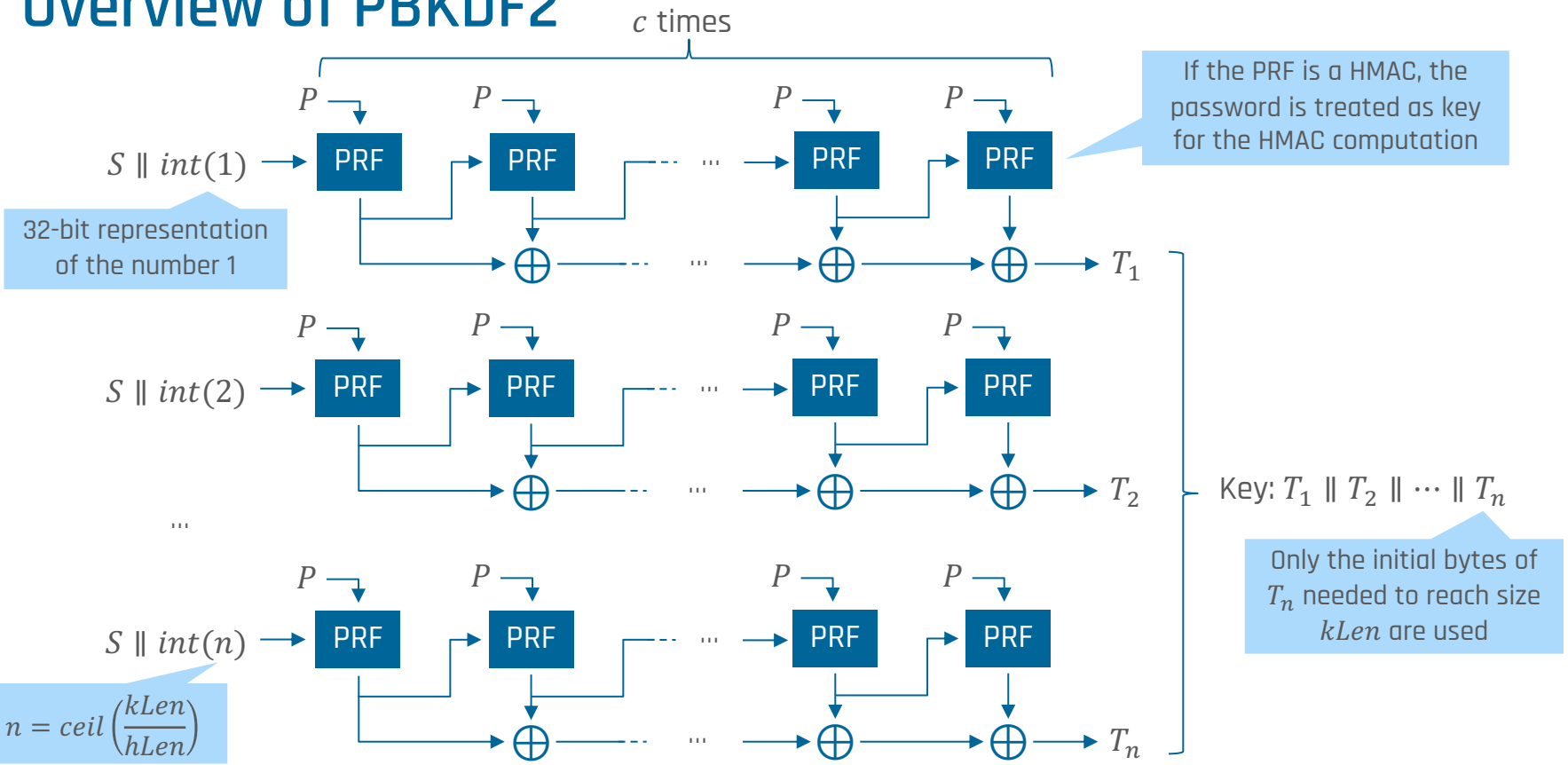
- Examples:
  - $\text{Hash} = h(h(h(\dots h(\text{Password} \parallel \text{Salt}) \dots)))$
  - $\text{Hash} = h(h(h(\dots h(\text{Password} \parallel \text{Salt}) \parallel \text{Salt} \dots) \parallel \text{Salt}) \parallel \text{Salt})$

Password-based Key Derivation Functions - PBKDF2

- PBKDF2 allows to derive a cryptographic key from a password
  - For password-based authentication, the key can be used as “hash” of the password

- Parameters of PBKDF2
  - A pseudorandom function PRF with two parameters and fixed output size $hLen$
    - A (keyed) HMAC is typically used
  - Password P
  - Salt S
  - Iteration count c
  - Desired key size $kLen$

Overview of PBKDF2



Other Key Derivation Functions

- PBKDF2 can be configured to arbitrarily increase **computation time**
  - However, it can be easily **parallelized** due to limited memory consumption
  - Specific integrated circuits (ASICs) and graphics cards (GPUs) can be used for highly-parallelized, offline attacks

- Modern functions (scrypt, Argon2, yescrypt) allow to configure **memory usage**
  - A **large** array, whose computation is **time intensive**, is used to derive the key
    - Elements are accessed depending on the password characters
  - Parallelisation of offline attacks requires either **huge memory** availability (for array storage) or **large computation time** (for recomputing array elements on the fly)

Password Storage in UNIX/Linux Systems

- User's credentials are stored in the file `/etc/shadow`
  - One line per user, 9 fields separated by :
  - First field is the username, second one contains password data

- Typical format of the password field: `$id$rounds=xxx$salt$hashedpwd$`
  - `id`: identifies the **algorithm** used to compute `hashedpwd`
    - Examples: 1: MD5, 5: SHA-256, 6: SHA-512, y: yescrypt
  - `rounds=xxx`: optional, considered only for SHA-256 and SHA-512
    - `xxx` is the **iteration count** to slow down offline attacks
    - 5000 iterations if missing
  - `salt`: a randomly generated **salt**, different for every user
  - Additional fields are present when yescrypt is used

```
mauro:$6$Wyb.F7nfn4qtq7mr$tTCSZVu7t5o/HsiFAiibc6493S9QYpcABfVPfI33GP
6Mh77ysg5l4Q/zwvf4ZTNEFf.b4P4MACT76uRGqEekv1:18738:0:99999:7:::
```

Further Countermeasures Against Online Attacks

- The attacker performs **many login requests** during an online attack
  - Real users, instead, generally enter the correct password by the third login attempt

- Login mechanisms can implement **reactive measures** to protect against such (automated) attacks
  - Enforcing a **delay** after wrong attempts
    - Initially small, grows with the number of attempts
  - **Locking the account** after a certain number of wrong guesses
    - Might be exploited to perform DOS attacks
  - Require a proof that requests are performed by a **human** (e.g., by solving a CAPTCHA)
  - Require **multi-factor authentication** (more later)

Strategies for Good Password Security

- Use appropriate **length** and **alphabet**
  - NIST suggests passwords not shorter than **8 characters** (the longer, the better)
  - Allowed characters should include **at least** digits, lower- and uppercase letters
  - Helps preventing **brute-force attacks**
- Do not use a **single, common word**
  - Helps preventing **dictionary attacks**
- Use **different passwords** for different services
  - Stealing the password of a service **does not** affect the security of the others!
  - An attacker might even **operate** one of the services where you are registered!
- Use **password managers** to store passwords!

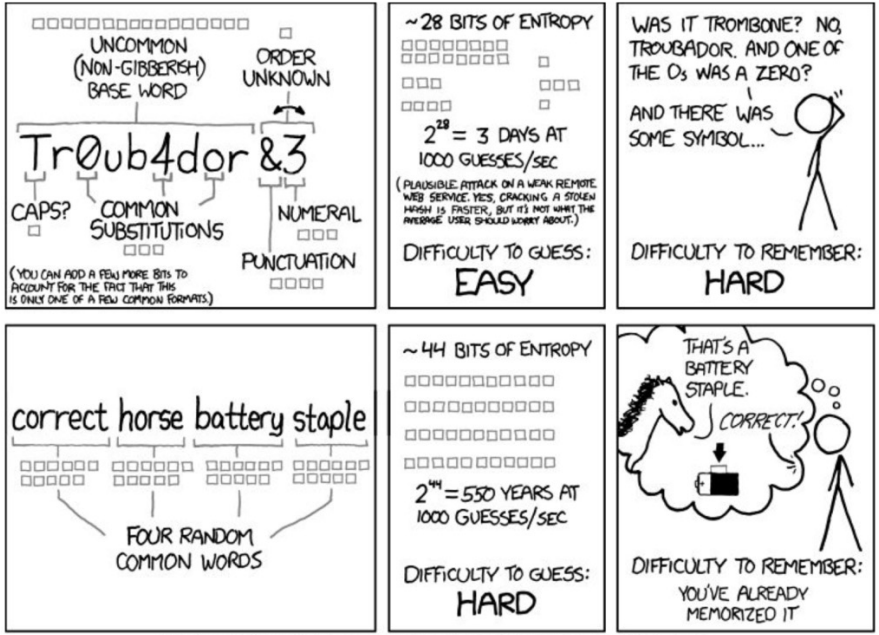
Most Common Passwords in Data Breaches (2023)

Ranking	Password	Usage Counter
1	123456	4.524.867
2	admin	4.008.850
3	12345678	1.371.152
4	123456789	1.213.047
5	1234	969.811
6	12345	728.414
7	password	710.321
8	123	528.086
9	Aa123456	319.725
10	1234567890	302.709

Ranking	Password	Usage Counter
11	1234567	234.187
12	123123	224.261
13	111111	191.392
14	Password	177.725
15	12345678910	172.502
16	000000	168.653
17	admin123	159.354
18	111	144.262
19	P@ssw0rd	135.424
20	root	122.834

https://nordpass.com/most-common-passwords-list/

Remembering Passwords



Password Managers

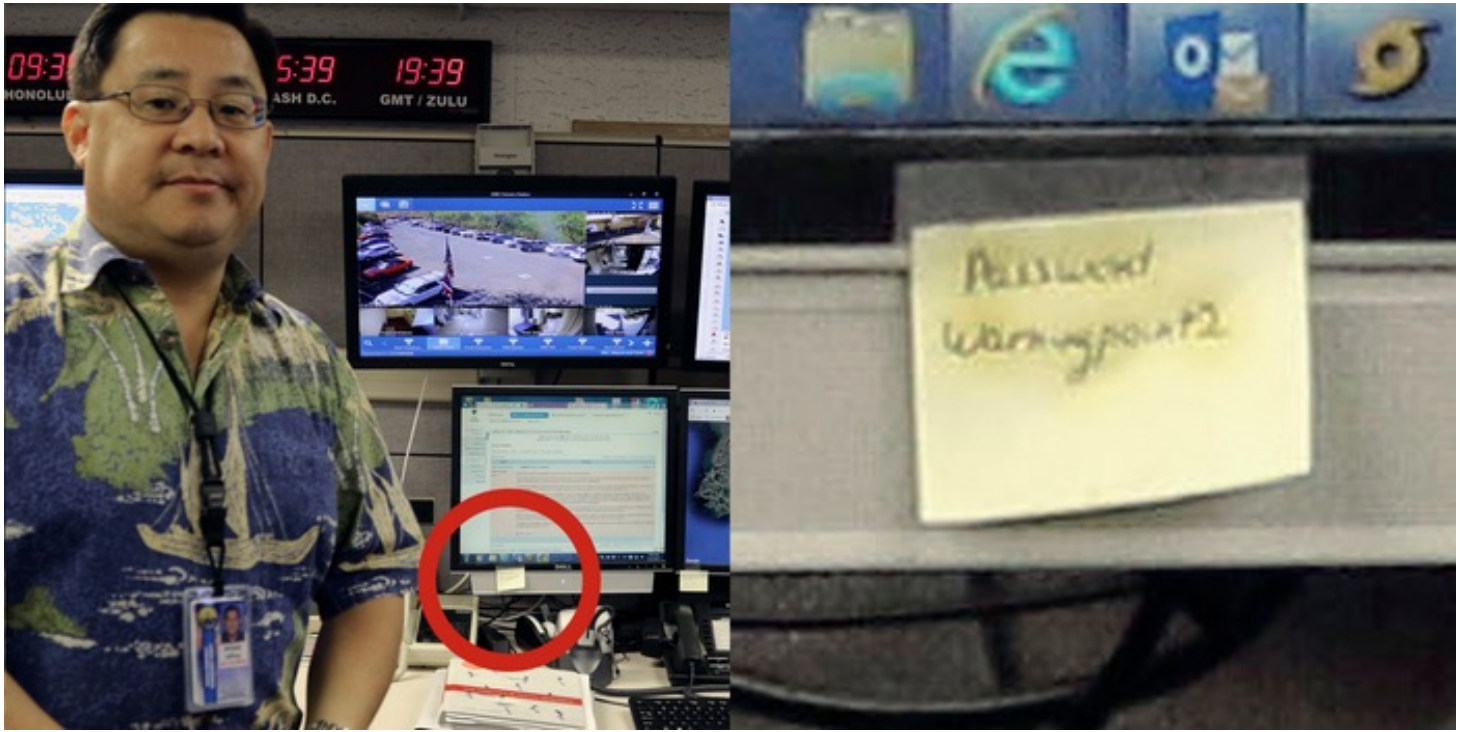
Desktop Applications

- Standard applications running on the user's machine
- Passwords are stored in encrypted form on a file **on the user's machine**
  - Encryption key derived from a master password
- Usage on multiple systems is not trivial
  - Passwords file must be copied on all used systems
  - Synchronisation needed after every file update
- Example: KeePass

Cloud-based Password Managers

- Both standard application and web-based versions are typically available
- Passwords are stored in encrypted form **on the systems** of the password manager's **developers**
  - Encryption key derived from a master password
- Synchronisation happens automatically!
- **Security guarantees strongly depend on the specific solution**
- Examples: 1Password, Bitwarden, Dashlane

A Post-It is not a Good Password Manager ☺



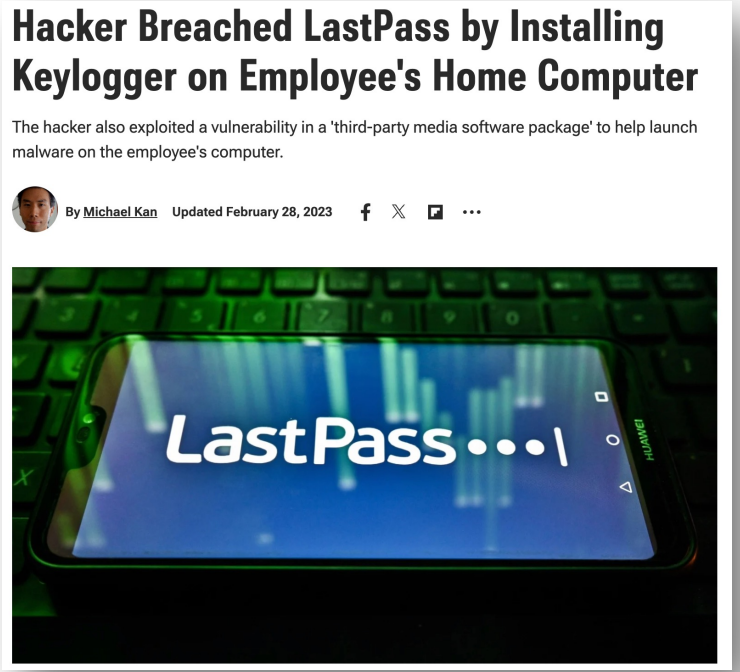
Password Theft – Shoulder Surfing

- Shoulder surfing denotes the practice of observing the execution of a login in a public area to discover the credentials
  - Usage of smartphone in parks or trains
  - Computer monitor in your room visible from a public street
  - Withdrawing cash from an ATM
  - Paying with credit card in shops

Password Theft - Keyloggers

- Keyloggers **record all keystrokes** tipped by a user on the keyboard
  - Goal is to capture sensitive user data, such as passwords

- Collected data is periodically sent to **drop zones** available on Internet
  - Attackers can connect to a drop zone to download undiscovered all leaked data



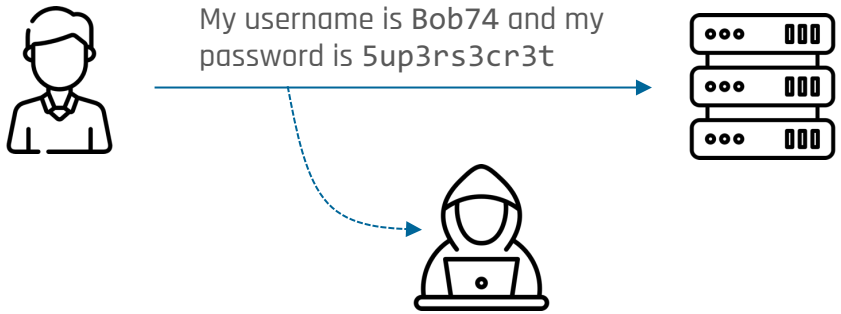
Password Theft - Phishing

- Scam where an attacker pretends to be a trustworthy entity to tricks the victim into revealing sensitive data (e.g., credentials)
  - Involves direct communication with the victim (e-mail, phone, instant messengers)
  - Victims are often directed to websites that are (almost) indistinguishable from original ones

- Classes of phishing
  - Classic: huge number of messages spammed to as many recipients as possible
  - Spear phishing: targets are accurately researched, and communication is tailored to the victim to make it more plausible

Password Theft – Network Attacks

- Passwords transmitted in **cleartext** can be trivially intercepted by a network attacker positioned between the user and the target server
  - Credentials must be always transferred over **encrypted channels** (e.g., TLS)



Data Breaches

AT&T acknowledges data leak that hit 73 million current and former users

Data leak hit 7.6 million current AT&T users, 65.4 million former subscribers.

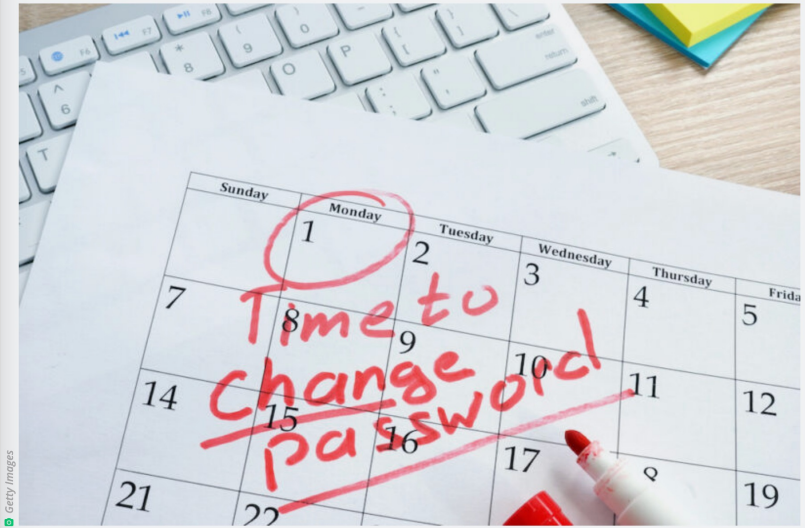
JON BRODKIN - 4/1/2024, 9:01 PM

Getty Images | VIEW press

Researcher uncovers one of the biggest password dumps in recent history

Roughly 25 million of the passwords have never been seen before by widely used service.

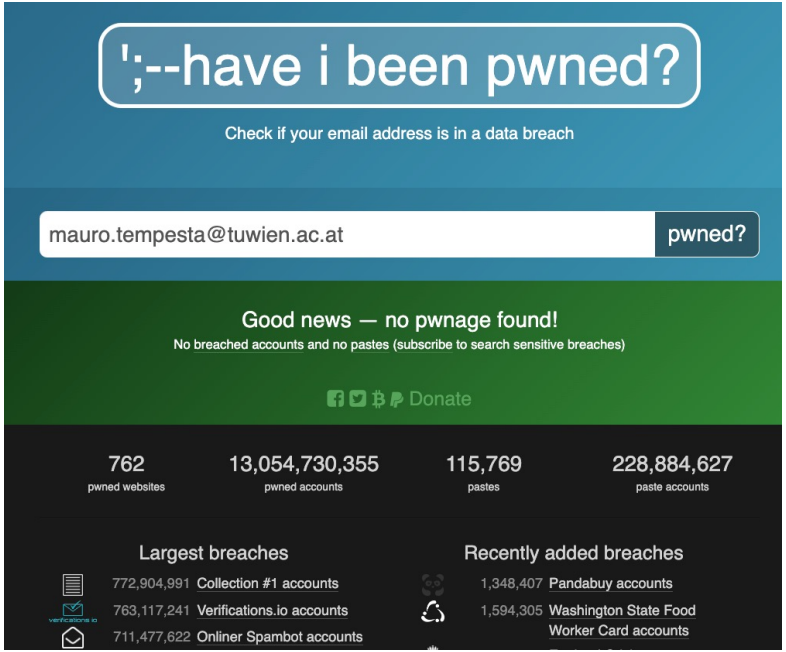
DAN GOODIN - 1/18/2024, 1:04 AM



Getty Images

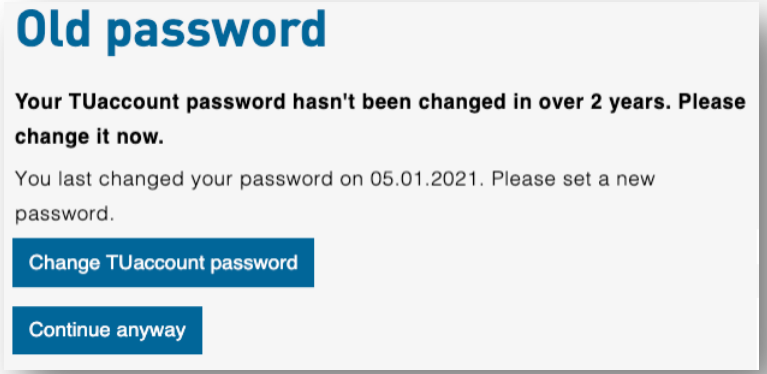
Have I Been Pwned?

- https://haveibeenpwned.com
  - Check for data breaches involving a specific **e-mail address**
  - Check if a specific **password** appears in data breaches
    - Provided password is hashed in the browser
    - **First 5 digits** of the hash value (SHA-1, hex format) are used to query for hashes of leaked passwords starting with that prefix
    - Search of the full hash performed via JavaScript in the browser



Password Rotation

- Some systems suggest / enforce a **password change** after a certain time span
  - Goal is to **reduce the likelihood** of successful brute-force attacks by reducing the lifetime of passwords



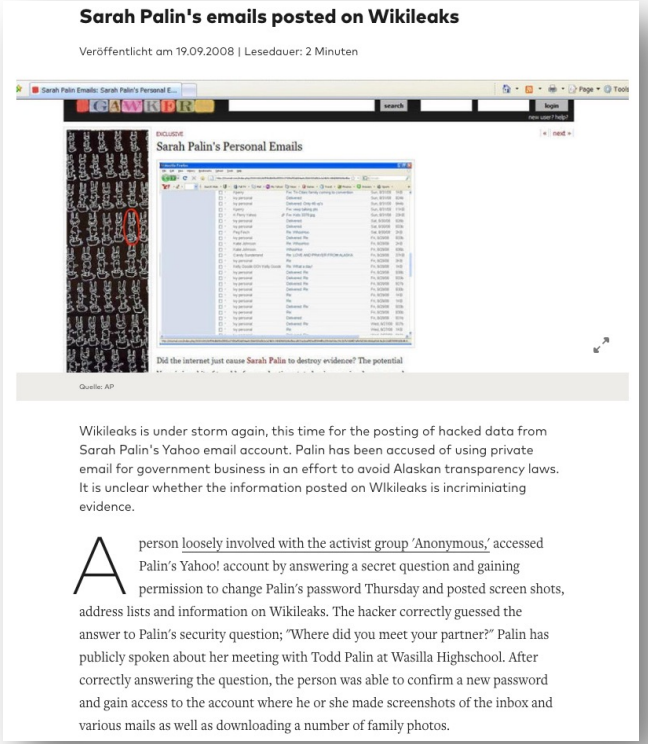
- Issue: Good passwords are **hard to remember**!
  - Typical users do not make use of password managers
  - If forced to completely change the password, they tend to write it down (e.g., post-it)

- NIST guidelines advise **against** the enforcement of password rotation

(Security) Questions for Password Recovery

- Some websites allow to **reset** an account's password after answering some *security* questions
  - Questions typically regard the **user's life**
    - Surname of teacher at school
    - Mother's surname before the wedding
  - Answers are provided when creating the account

- Such questions can be easily answered by strangers with some **social engineering** or by checking **social networks**
  - If possible, this mechanism should be **deactivated**!



Sarah Palin's emails posted on Wikileaks

Veröffentlicht am 19.09.2008 | Lesedauer: 2 Minuten

Wikileaks is under storm again, this time for the posting of hacked data from Sarah Palin's Yahoo email account. Palin has been accused of using private email for government business in an effort to avoid Alaskan transparency laws. It is unclear whether the information posted on WIkileaks is incriminiating evidence.

A person loosely involved with the activist group 'Anonymous,' accessed Palin's Yahoo! account by answering a secret question and gaining permission to change Palin's password Thursday and posted screen shots, address lists and information on Wikileaks. The hacker correctly guessed the answer to Palin's security question; "Where did you meet your partner?" Palin has publicly spoken about her meeting with Todd Palin at Wasilla Highschool. After correctly answering the question, the person was able to confirm a new password and gain access to the account where he or she made screenshots of the inbox and various mails as well as downloading a number of family photos.

One-Time Passwords and Challenge-Response Methods

One-Time Passwords (OTP)

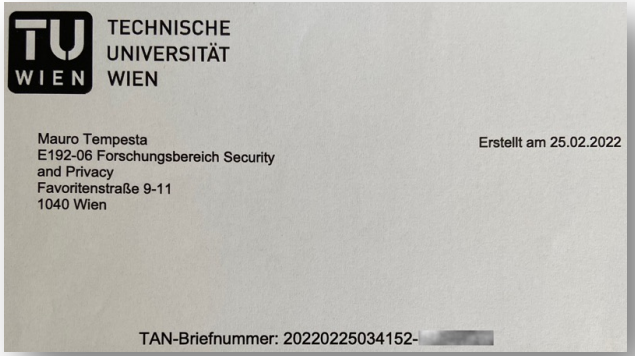
- Authentication methods in which a password can be used only **once**
  - Prevent **replay attacks**, issues if the first usage can be **intercepted** by an attacker
  - Passwords can be **precomputed** (TAN lists) or generated **on demand** (hardware tokens, generator apps)

- Typically employed only for security-critical applications
  - Often used, together with other schemes, as part of a **multi-factor authentication**

One-Time Passwords - TAN Lists

- Passwords are precomputed and communicated to the user over a **secure channel** (e.g., per post)

- Typically used as a **second factor** to confirm the execution of security-critical operations
  - User might have to provide a specific OTP or can provide any still unused password from the list
  - Number of executable operations is limited by the size of the passwords list



TECHNISCHE UNIVERSITÄT WIEN

Mauro Tempesta
E192-06 Forschungsbereich Security and Privacy
Favoritenstraße 9-11
1040 Wien

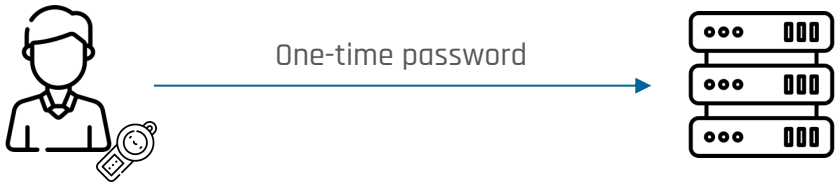
Erstellt am 25.02.2022

TAN-Briefnummer: 20220225034152-[redacted]

Nr.	TAN	Nr.	TAN	Nr.	TAN	Nr.	TAN	Nr.	TAN	Nr.	TAN
1	165054	31	685033	61	225204	91	005050	121	289358	151	316455
2	845507	32	146500	62	930462	92	371251	122	196743	152	591789
3	688850	33	507060	63	001353	93	174368	123	690501	153	063157
4	506509	34	806187	64	969211	94	255887	124	267638	154	998327
5	463462	35	570485	65	507175	95	698941	125	125785	155	963917
6	972181	36	178959	66	954827	96	412793	126	947126	156	173673
7	510260	37	311061	67	860843	97	346604	127	361607	157	510586
8	811245	38	142901	68	449222	98	304109	128	835859	158	847480
9	528381	39	541812	69	612733	99	178815	129	667668	159	886215
10	354380	40	842795	70	877681	100	186211	130	091782	160	360471
11	685503	41	905695	71	190583	101	252128	131	150781	161	046297
12	149190	42	340713	72	013089	102	010525	132	388425	162	015563
13	233634	43	120138	73	538729	103	107691	133	327464	163	423939
14	271472	44	500192	74	660682	104	427311	134	789149	164	212198
15	083584	45	394692	75	591211	105	072846	135	450429	165	377554
16	781652	46	952066	76	142073	106	246700	136	115829	166	702449
17	057563	47	632726	77	078214	107	034065	137	270625	167	000129
18	010308	48	657805	78	132441	108	463484	138	386727	168	031698
19	047607	49	892735	79	992048	109	819562	139	514198	169	064698
20	089122	50	424391	80	177199	110	266456	140	885798	170	250682
21	057189	51	051256	81	926733	111	668943	141	541133	171	219148
22	275729	52	735429	82	649333	112	715384	142	927297	172	054624
23	760516	53	062270	83	979334	113	555818	143	851729	173	953267
24	555938	54	168466	84	700794	114	595121	144	819699	174	000645
25	358098	55	262016	85	809974	115	787630	145	817963	175	299605
26	283196	56	303206	86	849977	116	706220	146	364162	176	381250
27	296369	57	982402	87	513173	117	153356	147	756114	177	130486
28	483145	58	908005	88	153377	118	407568	148	504750	178	048198
29	322956	59	574082	89	546104	119	572795	149	936409	179	846012
30	036833	60	595195	90	043546	120	525391	150	314965	180	002058

One-Time Passwords – OTP Generators

- The user and the authentication server compute the password **on the fly** during the authentication process
  - **Cryptographic algorithms** (hash functions, encryption schemes) are used to generate **short-lived** OTPs
  - Value of the OTP generally depends on the **time** in which it was generated
  - On the user's side, the OTP is computed on **specialised hardware** (hardware tokens) or by **generator apps** (e.g., installed on the smartphone)



OTP Generators – Computation and Validation

- Possible implementations
  - Hash functions: $\mathrm{OTP} = h(\mathrm{Time} \parallel S_X)$
  - Symmetric encryption: $\mathrm{OTP} = Enc(S_X, \mathrm{Time})$

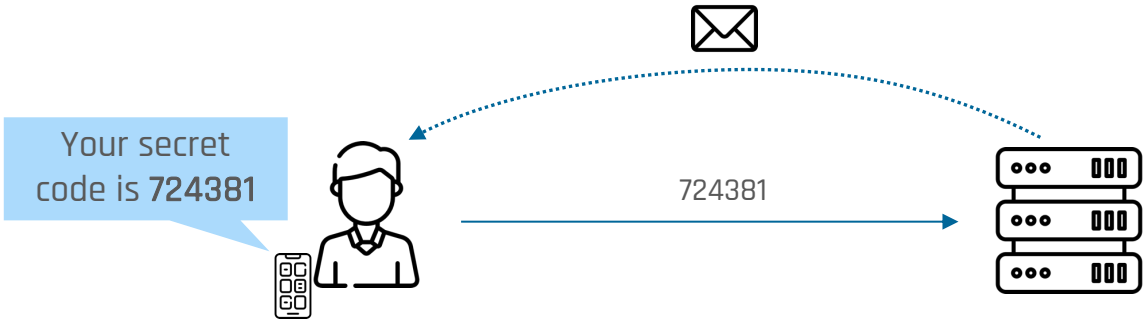
User-specific secret

- Both user and authenticating server need to know the user-specific secret S_X
  - To prevent storing a secret for every user, S_X is generated by combining some **user-specific** information and a **master key** (e.g., hash of concatenation of username and key)

- Validation
  - The server computes **multiple OTPs** (e.g., those generated in the last 5 minutes)
  - Success if **any** of these matches the OTP provided by the user

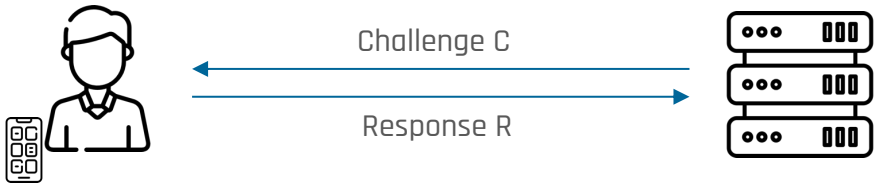
One-Time Passwords – Mobile TAN

- An OTP is sent to **via SMS** to the phone number of the user
  - Phone number is provided during the creation of the account
    - Number can be changed only after proving the possession of the previous phone number or by contacting the technical support
  - Knowledge of the OTP proves the **possession of the SIM** card of the user



Challenge-Response Methods

- Similar to OTP generators, but the OTP depends on some **server's generated data**
  - Authentication server sends a **challenge C** to the user
    - Value C must include some **randomly generated component** (prevention of replay attacks)
    - Value C can partially depend on the **operation** to be authorized
  - User performs a **cryptographic operation** on C to compute the **response R**
    - Response is usually computed by an **authenticator app** (e.g., installed on the smartphone or PC of the user)
    - Ability to compute R proves the **possession** of the required secret / cryptographic key



Variations of Challenge-Response Methods

- **Hash computation** based on shared secret S_X

$$R = h(C \parallel S_X)$$

- **Symmetric encryption** (e.g., with AES) using a shared, secret key K_X

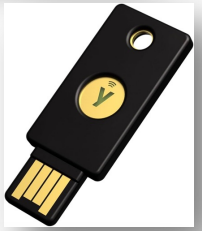
$$R = Enc(K_X, C)$$

- **Signature computation** (e.g., with RSA) using the private key SK_X

$$R = Sign(SK_X, C)$$

Hardware Authentication Tokens

- **Disconnected tokens**
  - Authentication data shown on integrated display
  - Usable as **OTP generators**
- **Connected tokens**
  - Must be plugged to the user's device (via USB, NFC, Bluetooth, dedicated card reader)
  - Can be used as **OTP generators** or for **challenge-response** authentication
  - **Web browsers** can use them for online authentication (via the WebAuthn API)



Hardware Tokens – Advantages and Disadvantages

Advantages

- No need to remember any authentication information (e.g., password) if used as unique factor
- Limits the effect of password reuse when used for multi-factor authentication

Disadvantages

- Need to remember to bring the device with you all the time
- Can be relatively expensive
  - Up to 30-50€ depending on the type
- Complete loss of security if the device is lost or stolen
  - Unless used for multi-factor authentication

Biometrics

Biometrics

Biometric systems employ **biological** or **behavioural** characteristics of users for authentication purposes

Passive	Active
Fingerprint	Voice recognition
Face recognition	Keystroke dynamics
Retina / iris recognition	Signature recognition
Hand geometry	Lips movement (while speaking)
Palm veins	Mouse movement
DNA	Gait analysis

Desired Characteristics of Biometric Traits

- **Universality**
  - Every user of the system must possess the trait
- **Uniqueness**
  - The trait should be sufficiently different between individuals such that it can be used to distinguish one individual from the others
- **Measurability**
  - The trait must be measurable in a quantitative way
  - Data can be collected in a form that permits subsequent processing and extraction of the relevant features

Desired Characteristics of Biometric Traits

- **Permanence**
  - The trait should not vary too much over time
  - Matching algorithm should be resistant to small measurement changes
- **Unforgeability**
  - The trait should be difficult to forge / imitate
- **Performance**
  - The trait should be quick, accurate and efficient to measure / compare

Error Measures

- **Deviations** must be taken into account when comparing biometric traits
  - Face recognition: glasses, beard, haircut
  - Fingerprint: sweating, dirt on the reader
- This **tolerance** may lead to mistakes (**false positives** and **false negatives**)

False Acceptance Rate (FAR)

- Measures the security of the system
  - How many unauthorized users are given access to the system?

$$FAR = \frac{\text{Number of false acceptances}}{\text{Total number of impostor attempts}}$$

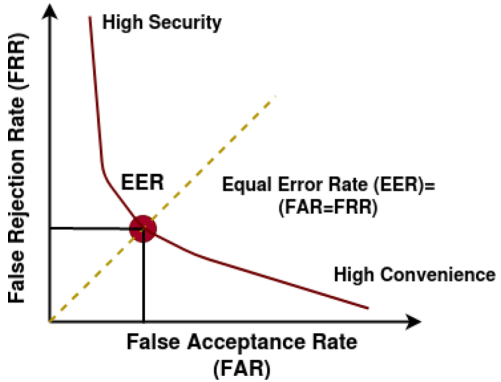
False Rejection Rate (FRR)

- Measures the usability of the system
  - How many authorized users are denied access to the system?

$$FRR = \frac{\text{Number of wrong rejections}}{\text{Total number of legitimate attempts}}$$

Error Measures

- The **equal error rate** represents a good compromise between security and usability
  - The tolerance value is chosen so that FAR = FRR
- A different value can be chosen depending on the actual security and comfort requirements

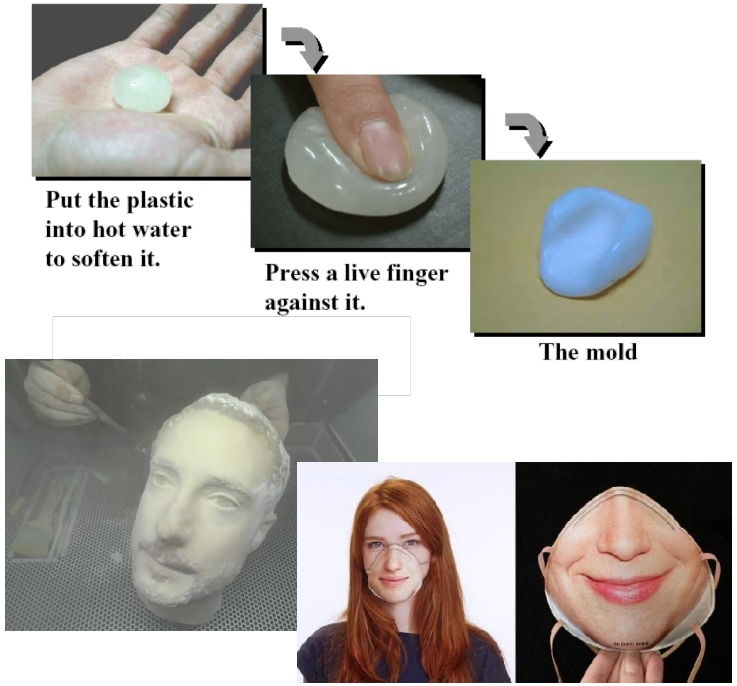


Biometric feature	Typical FAR (in %)	Typical FRR (in %)
Fingerprint	0,001 ... 2	0,1 ... 5
Iris recognition	0,0001 ... 1	0,1 ... 2
Face recognition	0,5 ... 2	1 ... 3
Hand geometry	1 ... 4	1 ... 5

Attacks Against Biometrics

- Many attack techniques against biometrics have been successful over the years
  - Face recognition: usage of photos, masks, 3D reconstructions
  - Fingerprint: creation of artificial fingers with stolen fingerprint

- **Liveness detection** can counter such attacks
  - Fingerprint: measuring blood pressure / flow
  - Iris scanning: analysis of pupil movement



Biometric Methods – Advantages and Disadvantages

Advantages

- No need to **remember** any authentication information (e.g., password) if used as unique factor
- Authentication information **cannot** be forgotten or lost

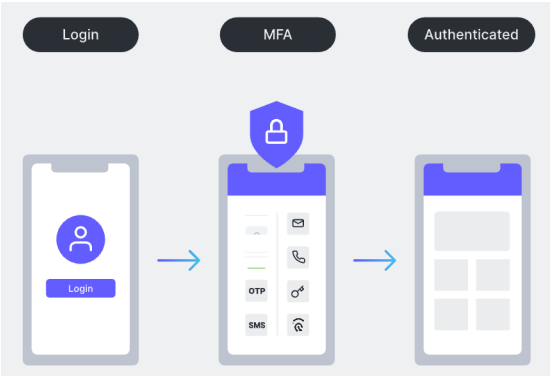
Disadvantages

- Generally inapplicable on the Internet
  - **Privacy issues**: users are not willing to share biometric data online
- **Reliability problems**
  - Bypasses due to different users with similar biometric traits
- **Revocation** is difficult / impossible
  - What to do in case of forging?

Multi-Factor Authentication

Multi-Factor Authentication

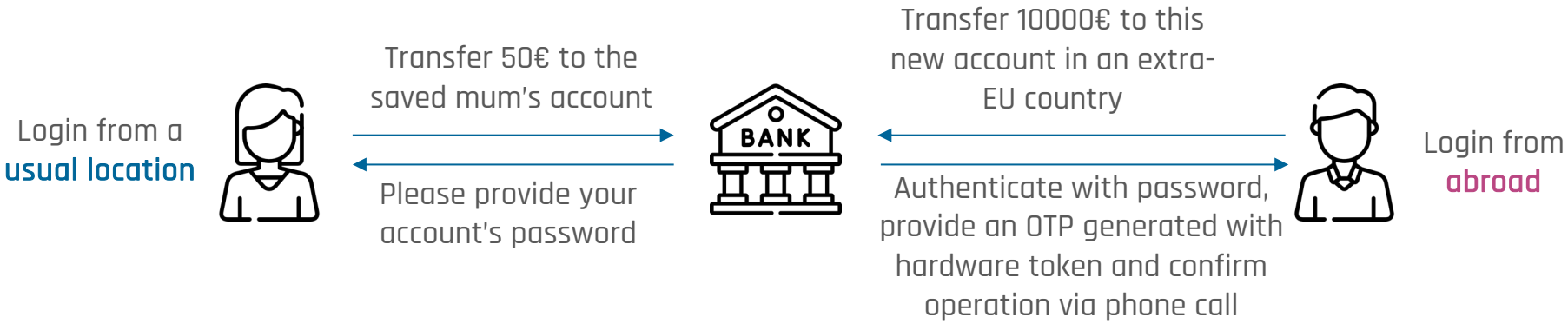
- Multiple authentication methods from **different classes** (knowledge, ownership, inherence) are employed together for user authentication
  - Goal is to overcome the limits of an authentication method via the others



- Examples
  - Password (knowledge) + OTP generated with hardware token (ownership)
  - Face recognition (inherence) + authenticator app installed on the phone (ownership)
  - Password (knowledge) + OTP from a TAN list (ownership) + fingerprint (inherence)
  - **Password (knowledge) + PIN (knowledge) is not a multi-factor authentication!**

Risk-Based Authentication

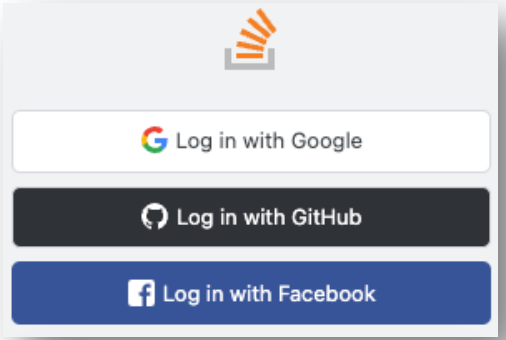
- Necessary authentication steps are chosen **depending** on various factors
  - Trustworthiness of the user
  - Criticality of the required operation
  - Conditions of the current access (e.g., location)
  - Habits of the user



Single Sign-On

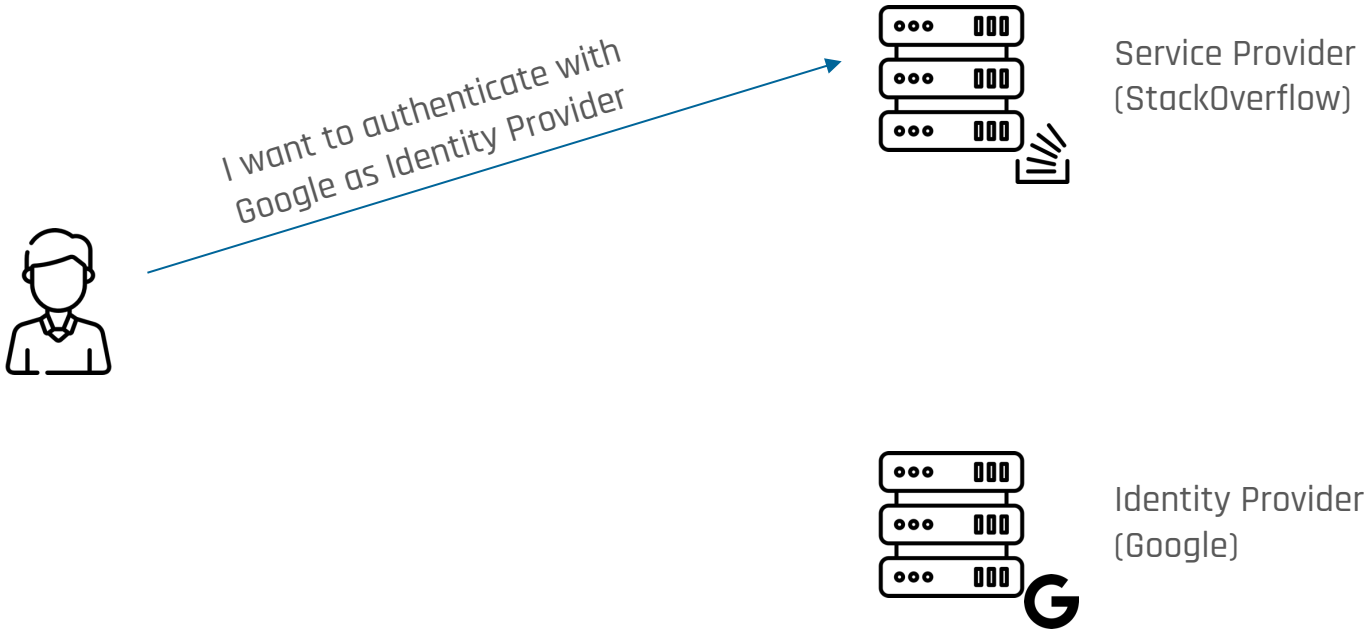
Single Sign-On

- Single Sign-On allows users to authenticate on multiple systems (**Service Providers**) by delegating the actual authentication process to an external **Identity Provider**

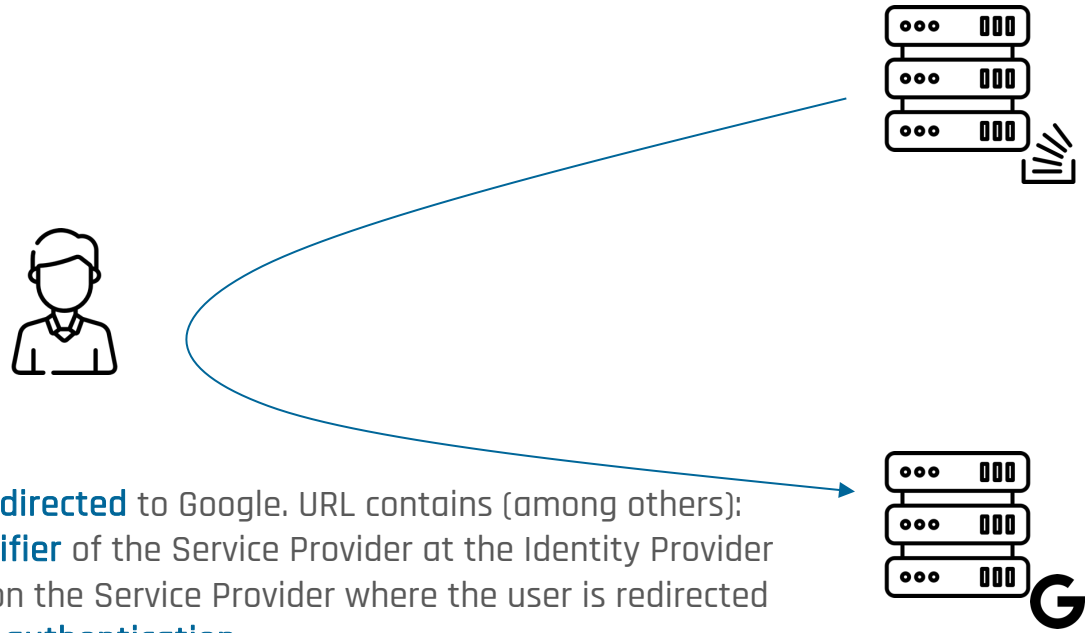


- Communication between user, service and identity provider is regulated by **standardised protocols**
  - Common choices are **OpenID Connect** (based on OAuth 2.0) and SAML
  - Protocols define multiple **flows** to handle different use cases
    - OpenID Connect: **Authorisation Code Flow** (classic web and mobile applications), **Implicit Flow** (web applications without backend), **Hybrid** (combination of the two, rarely used)

Overview of Authorisation Code Flow in OpenID Connect



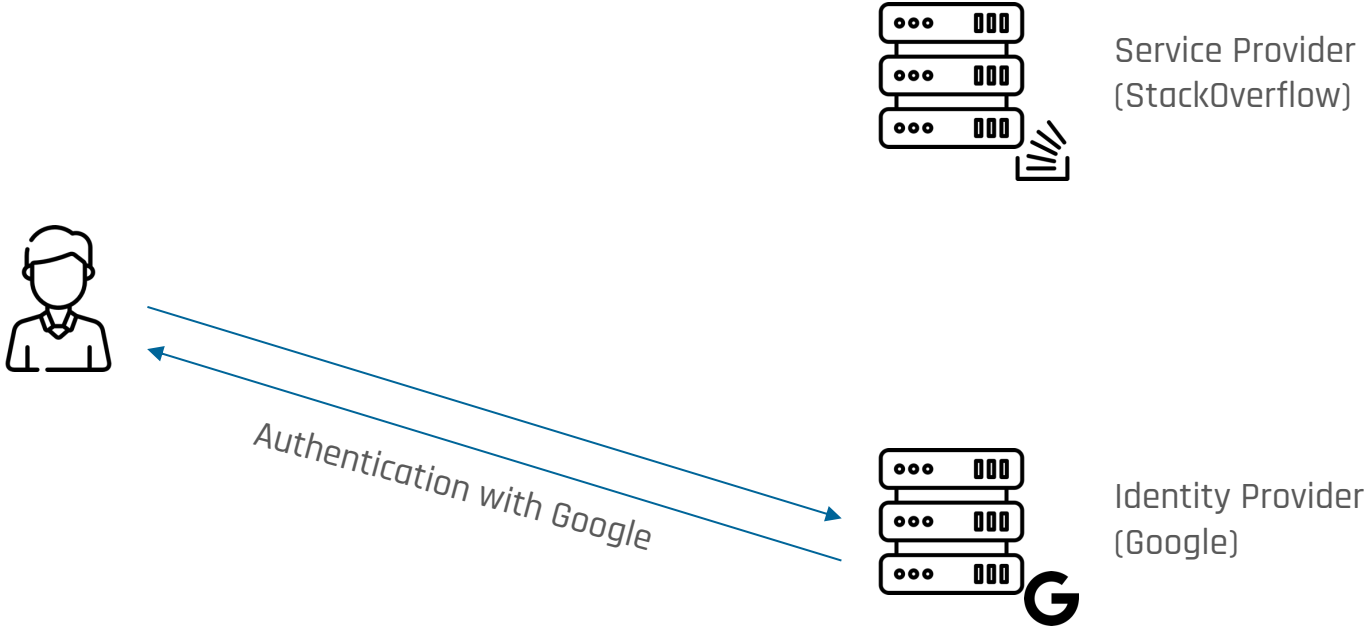
Overview of Authorisation Code Flow in OpenID Connect



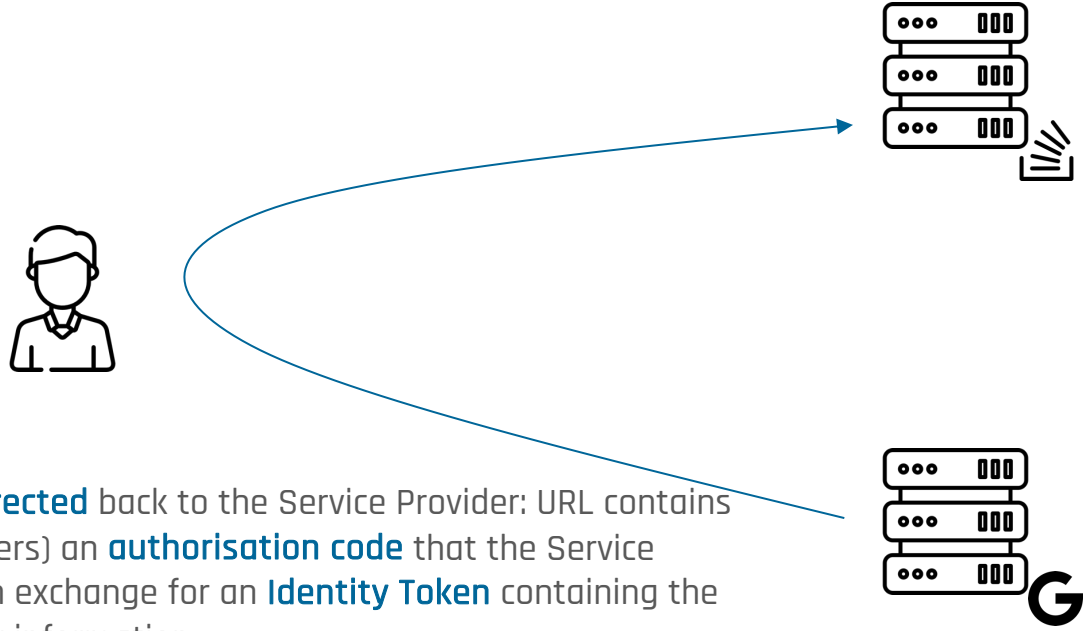
User is **redirected** to Google. URL contains (among others):

- **Identifier** of the Service Provider at the Identity Provider
- **URL** on the Service Provider where the user is redirected **after authentication**

Overview of Authorisation Code Flow in OpenID Connect



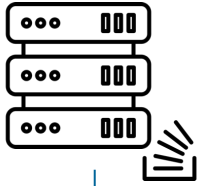
Overview of Authorisation Code Flow in OpenID Connect



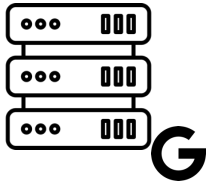
User is **redirected** back to the Service Provider: URL contains (among others) an **authorisation code** that the Service Provider can exchange for an **Identity Token** containing the desired user information

Overview of Authorisation Code Flow in OpenID Connect

Service Provider **authenticates** at the Identity Provider and sends the received **authorisation code**



Service Provider
(StackOverflow)



Identity Provider
(Google)

Overview of Authorisation Code Flow in OpenID Connect


```
{
  "iss": "https://idp.google.com",
  "sub": "g34nth543",
  "aud": "nbj593feD",
  "exp": 1311281970,
  "iat": 1311280970,
  "email": "aschmitt@gmail.com",
  "given_name": "Andreas",
  "family_name": "Schmitt"
}
```


Identity Provider **validates** the received code and issues an **identity token** for the user

Service Provider (StackOverflow)

Identity Provider (Google)

Advantages and Disadvantages of Single Sign-On

- Advantages
  - User credentials are **only stored** at the Identity Provider
  - Users need to **remember only** the credentials for the Identity Provider
  - Service Providers do not have to implement their own authentication methods
  - Identity Providers can offer multiple ways to authenticate

- Disadvantages
  - Identity Providers represent a **single point of failure**
    - Login on Service Providers not possible in case of availability issues at the Identity Provider
    - Theft of the credentials of the Identity Provider compromises security of all the accounts of the Service Providers for which the user employs Single Sign-On