

# OUTLINE

1

Basis of authentication:

- what you know, what you possess, what you are.

2

Password-related techniques

3

Attacks on passwords and defense mechanisms

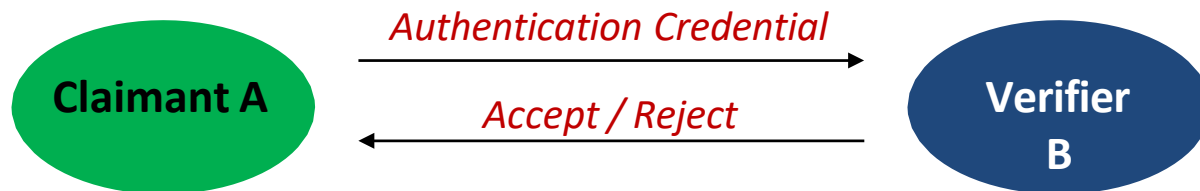
4

Authentication tokens and biometrics



## Entity Authentication

### Illustration



A *process* whereby one party (*verifier*) is assured of the identity of a second party (*claimant*) in protocol.

## Objectives of Authentication

1. Honest participant A (claimant) is able to successfully authenticate itself to B (verifier), i.e., B accepts A's identity.
2. Transferability: B cannot reuse an identification exchange with A to impersonate A to a third party C.
3. Impersonation: The probability that a third party C, distinct from A, playing the role of A, can cause B to accept A's identity is negligible.

# BASIS OF AUTHENTICATION

## What you know



- Authentication is done by exhibiting knowledge of certain secrets.
- Examples: passwords, Personal Identification Numbers (PINs), private/secret keys.

## What you have



- Magnetic/smart cards, hardware tokens (password generators).
- Typically combined with passwords, to form a two-factor authentication.

## Who you are



- Physical characteristics, e.g., fingerprints, voice, retinal patterns.
- Behavioral characteristics, e.g., handwritten signatures, keystroke dynamics.

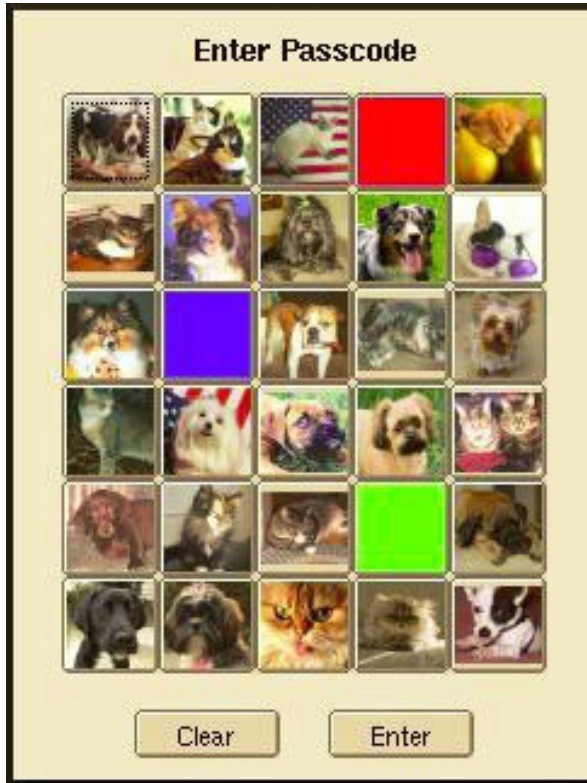
## Weak/Simple Authentication:

- Password-based.
- Unilateral: one entity (claimant) proves its identity to the verifier.
- Prove knowledge of secret by giving up the secret

## Strong Authentication:

- Involves mutual authentication; both parties take both the roles of claimant and verifier:
- Challenge-response protocols: sequence of steps to prove knowledge of shared secrets.
- Prove knowledge of secret WITHOUT giving up the secret

# PASSWORD-RELATED TECHNIQUES



- Password storage:
  - Plaintext (BAD) or “encrypted” (fair) or “hashed” (good).
- Password policies:
  - What rules need to be imposed on the selection of passwords by users, number of failed attempts, etc.
- “Salting” of passwords.
- Alternative forms of passwords
  - Passphrases, one-time passwords, visual passwords.

**Salt** is random data that is used as an additional input to a one-way function that “hashes” a password. Salts are used to safeguard passwords in storage. The primary function of salts is to defend against dictionary attacks.

Password storage security relies on a cryptographic construct called **one-way function**



**Hash functions** are an example of one-way function:

- A hash function  $f$  takes an input  $x$  of *arbitrary length*, and produces an output  $f(x)$  of *fixed length*.

A one-way function  $f$  is a function that is relatively **easy to compute** but **hard to reverse**.

- Given an input  $x$  it is easy to compute  $f(x)$ , but given an output  $y$  it is hard to find  $x$  so that  $y = f(x)$



# PROPERTIES OF HASH FUNCTIONS

Suppose  $H$  is a hash function. We say  $H$  satisfies:

- *Pre-image resistant* if given a hash value  $y$ , it is **computationally infeasible** to find  $x$  such that  $H(x) = y$ .
- *Collision resistant* if it is **computationally infeasible** to find a pair  $(x, y)$  such that  $x \neq y$  and  $H(x) = H(y)$ .

**Recap:** A one-way function  $f$  is a function that is very easy to compute but hard to reverse. Hash function is an example of one-way function. Impt Hash Functions: : **SHA256, 512, KECCAK (crypto)**, ARGON2, bcrypt (for password hashing)

# PASSWORD STORAGE

## Plaintext

- Passwords stored in plaintext.
- Claimant's password is checked against the database of passwords.
- **No protection against insider** (system admin) or an attacker who gains access to the system.  
Hence **dispute is possible!**

## Hashed/ encrypted passwords

- Passwords are encrypted, or hashed, and only the encrypted/hashed passwords are stored.
- Claimant's password is hashed/encrypted, and checked against the database of hashed/encrypted password.
- Some degree of protection against insider/attacker.



# PASSWORD STORAGE

In operating systems, password hashes are stored in a password file.



In Windows system, passwords are stored in Security Accounts Manager (SAM) file  
(%windir%\system32\config\SAM).

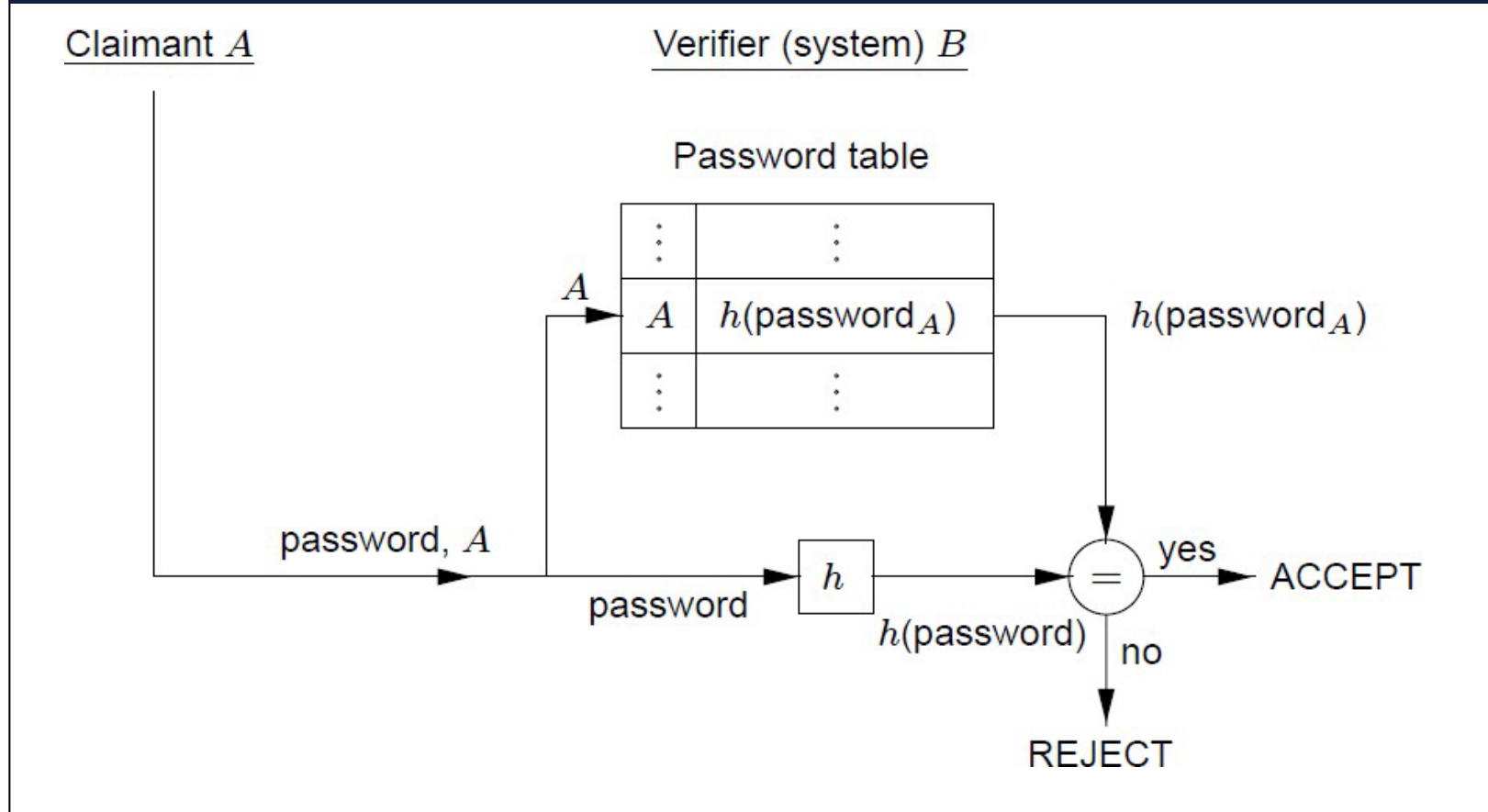


In Unix, this is `/etc/passwd`, but in modern Unix/Linux systems it is in the *shadow* file in `/etc/shadow`.

- At the application levels, passwords may be held temporarily in intermediate storage locations like buffers, caches, or a web page (don't save passwords in cache!)
- The management of these storage locations is normally beyond the control of the user; a password may be kept longer than the user has bargained for.

# HASHED PASSWORD VERIFICATION

Notice that the verifier **does (should) not** store the passwords, only their hashes



Source: Menezes et al. Handbook of Applied Cryptography.

# ATTACK ON PASSWORDS

## Offline Guessing Attacks



Exhaustive attacks  
Intelligent attacks: Dictionary attacks

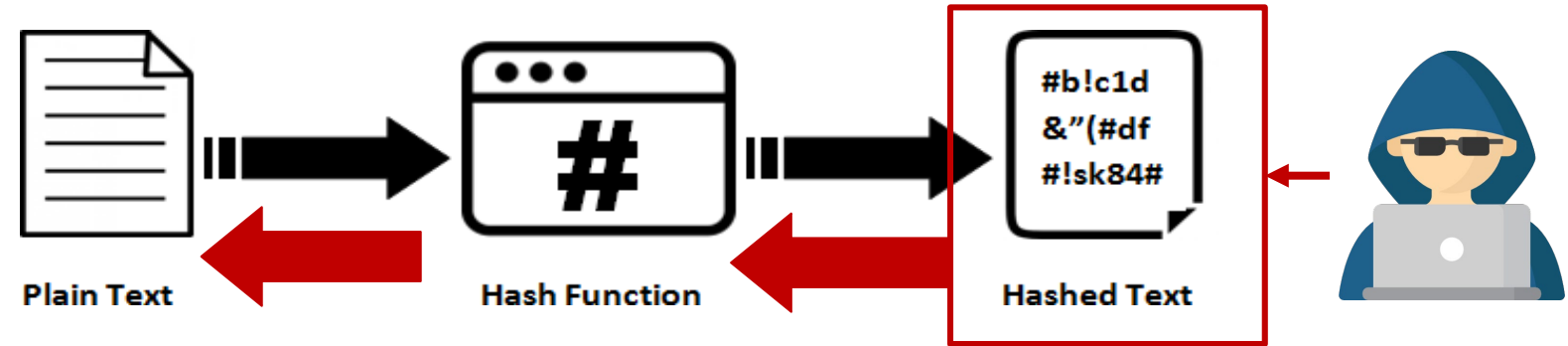
## “Phishing” and Spoofing



# OFFLINE GUESSING ATTACK

## Offline Guessing Attack

An attack where the attacker obtains the hashed passwords, and attempts to guess the passwords.



- This is a plausible threat, due to:
  - many incidents of **stolen (hashed) passwords** as a consequence of **hacks on servers** or **sniffing traffic**
  - usage of the **same passwords across different** accounts; so **compromise of a password for one account affects other accounts.**

**Recap:** In Unix, password hashes are stored in `/etc/passwd`, but in modern Unix/Linux systems it is in the *shadow* file in `/etc/shadow`.

# PASSWORD-RELATED INCIDENTS

## SingHealth cyber attack a result of human lapses, IT system weaknesses: COI report

By CYNTHIA CHOO



Reuters file photo

The SingHealth cyber attack happened because of lapses by employees and vulnerabilities with the system.

Published 10 JANUARY, 2019 UPDATED 10 JANUARY, 2019

85 Shares



SINGAPORE — The SingHealth cyber attack happened because of lapses by employees and vulnerabilities with the system. Ultimately, the breach into the public healthcare group's database was preventable even though the attacker was skilled.

These were the key findings in a report released on Thursday (Jan 10) by

**Vulnerabilities and weaknesses in the SingHealth network and SCM system contributed to the attacker's success in obtaining and taking the data**

- The SCM database, which is legally owned by SingHealth, functioned on an open network that was linked to the Citrix servers of Singapore General Hospital (SGH), which resulted in a critical vulnerability the attacker exploited.
- It was found that there was a lack of monitoring of the SCM database for unusual queries and access. For one, there was no existing control to detect or block bulk queries being made to the database. For another, the Citrix servers of SGH were not monitored for real-time analysis and alerts of vulnerabilities and issues arising from these servers.
- The Citrix servers were not adequately secured against unauthorised access. Notably, the process requiring 2-factor authentication (2FA) for administrator access was not enforced as the exclusive means of logging in as an administrator. This allowed the attacker to access the server through other routes that did not require 2FA.
- Another weakness which may have been exploited by the attacker included **weak administrator account passwords**. This was among others discovered during a test but the remediation process undertaken by IHiS was mismanaged and inadequate, and a number of **vulnerabilities remained** at the time of the cyber attack.



# PASSWORD-RELATED INCIDENTS



29  
Oct 13

## Adobe Breach Impacted At Least 38 Million Users



The recent data breach at **Adobe** that exposed user account information and prompted a flurry of password reset emails impacted at least 38 million users, the company now says. It also appears that the already massive source code leak at Adobe is broadening to include the company's **Photoshop** family of graphical design products.

AnonNews - Everything Anonymous  
[FBI] [RIP] [Forum] [404] [404] [404]

## 6.46 million LinkedIn passwords leaked online

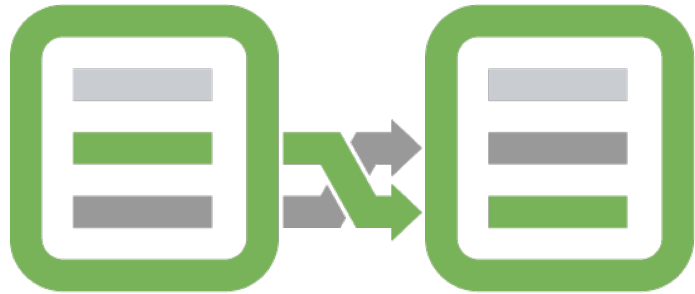
**Summary:** More than 6.4 million LinkedIn passwords have leaked to the Web after an apparent hack. Though some login details are encrypted, all users are advised to change their passwords.

By **Zack Whittaker** for **Between the Lines** | June 6, 2012 -- 05:46 GMT (13:46 SGT)



A vulnerability in Starbucks' mobile app could be putting coffee drinkers' information—including their usernames, email addresses and passwords—at risk.

# BRUTE FORCE ATTACK



MATCH

- Brute force guessing attack against passwords tries to guess password by enumerating all passwords and their hashes in sequence, and check whether they match the target hashes.
- A measure **against brute force attack is to increase the space of possible passwords**, e.g., longer passwords, allowing more varieties of symbols (alphabets, numerals, signs).

**Password policy is an important means to increase difficulties of brute force attack**

# PASSWORD ENTROPY-measured by $2^k$

$\rightarrow c$ $\downarrow n$	26 (lowercase)	36 (lowercase alphanumeric)	62 (mixed case alphanumeric)	95 (keyboard characters)
5	23.5	25.9	29.8	32.9
6	28.2	31.0	35.7	39.4
7	32.9	36.2	41.7	46.0
8	37.6	41.4	47.6	52.6
9	42.3	46.5	53.6	59.1
10	47.0	51.7	59.5	65.7

**Table 10.1:** Bitsize of password space for various character combinations. The number of  $n$ -character passwords, given  $c$  choices per character, is  $c^n$ . The table gives the base-2 logarithm of this number of possible passwords.

Source: Menezes et al. Handbook of Applied Cryptography.

At present, software password crackers can crack up to 16 million pswd/sec per pc.  
Write a program to calculate how long it will take to bruteforce  
passwords for each entry.



# PASSWORD ENTROPY-measured by $2^k$

$\rightarrow c$ $\downarrow n$	26 (lowercase)	36 (lowercase alphanumeric)	62 (mixed case alphanumeric)	95 (keyboard characters)
5	23.5	25.9	29.8	32.9
6	28.2	31.0	35.7	39.4
7	32.9	36.2	41.7	46.0
8	37.6	41.4	47.6	52.6
9	42.3	46.5	53.6	59.1
10	47.0	51.7	59.5	65.7

**Table 10.1:** Bitsize of password space for various character combinations. The number of  $n$ -character passwords, given  $c$  choices per character, is  $c^n$ . The table gives the base-2 logarithm of this number of possible passwords.

Source: Menezes et al. Handbook of Applied Cryptography.

Now  $2^{35}$  complexity can be cracked within **a day** on a 3GHz PC (generous est).

**1 FPGA Hardware cracker** can crack 56 bits within **5 days** (est).

ASIC crackers can be **more than 10 times faster** than FPGA.

- Choosing passwords with **high entropy** prevents brute-force attack.
- However, hashed passwords, especially for human-generated passwords, are still vulnerable to *dictionary attack*.
- This exploits weakness in human-chosen passwords, which tend to derive from words in natural languages.

**Users with same password will have same hash value stored in password file.**

- Guess some commonly used passwords
- Compute their hash values
- Look for the same hash values in the password file

## Strategy

A strategy for cracking hashed passwords is to **pre-compute a hash table**, containing pairs of passwords and their hashes.

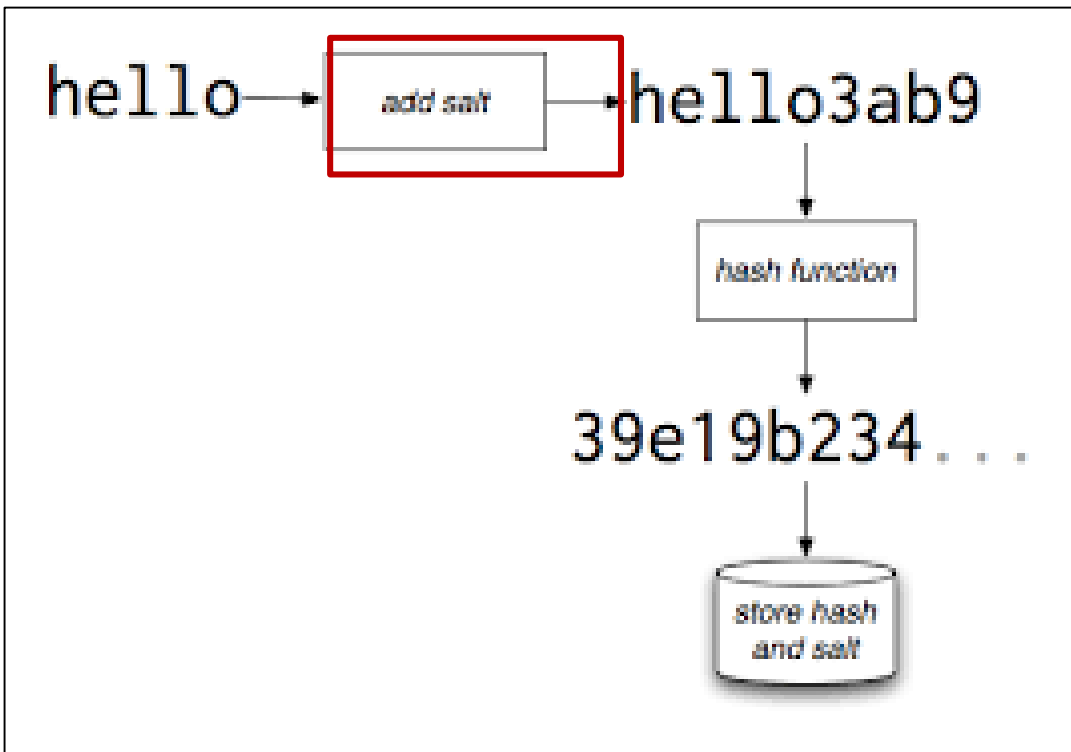
- If we have  $k$  password candidates and each hash has  $n$  bit, then we have a table of size  $k \times n$
- This may not be practical if  $k$  is large.

## Rainbow Table (not in quiz syllabus)

- Hash tables are often represented using a data structure called *rainbow table*.
  - Not all hashes are stored; some will be computed from the stored hashes.
  - Not all hashes are represented.
  - Tradeoff between space requirement and query time.

## Salting

### Illustration



- To reduce the effectiveness of offline attacks using pre-computed hashes, a *salt* is added to a password before applying the hash function.
- A salt is just a random string.
- Each password has its own salt.
- The salt value is stored along with the hash of password+salt.
- For a salt of  $n$ -bit, the attacker needs to pre-compute  $2^n$  of hashes for *the same password*.

# Password Storage Cheat Sheet

## Introduction📖

- It is essential to store passwords in a way that prevents them from being obtained by an attacker even if the application or database is compromised.
  - The majority of modern languages and frameworks provide built-in functionality to help store passwords safely.
- After an attacker has acquired stored password hashes, they are always able to brute force hashes offline.
- As a defender, it is only **possible to slow down offline attacks** by **selecting hash algorithms** that are as **resource intensive as possible**.

# Hashing vs Encryption

- Hashing and encryption both provide ways to keep sensitive data safe.
- **Passwords should be hashed, NOT encrypted.**
- **Hashing is a one-way function** (i.e., it is impossible to "decrypt" a hash and obtain the original plaintext value).
- Hashing is appropriate for password validation.
- Even if an attacker obtains the hashed password, they cannot enter it into an application's password field and log in as the victim.
- **Encryption is a two-way function**, meaning that the original plaintext password can be retrieved (if we have the key)

# How Attackers Crack Password Hashes

- Although it is not possible to "decrypt" password hashes to obtain the original passwords, it is possible to "crack" the hashes in some circumstances. The basic steps are:
- Select a password you think the victim has chosen (e.g.password1!)
- Calculate the hash
- Compare the hash you calculated to the hash of the victim.
- If they match, you have correctly "cracked" the hash and now know the plaintext value of their password.

# How Attackers Crack Password Hashes

- This process is repeated for a large number of potential candidate passwords.
- Different methods can be used to select candidate passwords, including:
  - Lists of passwords obtained from other compromised sites
  - Brute force (trying every possible candidate)
  - Dictionaries or wordlists of common passwords



# How Attackers Crack Password Hashes

- While the number of permutations can be enormous, with high speed hardware (such as GPUs) and cloud services with many servers for rent, the cost to an attacker is relatively small to do successful password cracking especially when best practices for hashing are not followed.
- **Strong passwords stored with modern hashing algorithms and using hashing best practices should be effectively impossible for an attacker to crack.**
- It is your responsibility as an administrator to select a modern hashing algorithm (later)

# Password Storage Concepts

- **Salting:**
- A salt is a **unique, randomly generated** string that is added to each password as part of the hashing process.
- As the salt is **unique** for **every user**, an attacker has to crack hashes one at a time using the respective salt rather than calculating a hash once and comparing it against every stored hash.
- This makes cracking large numbers of hashes significantly harder, as the time required grows in direct proportion to the number of hashes.

# Password Storage Concepts

- Salting also **protects against an attacker pre-computing hashes** using rainbow tables or database-based lookups.
- Finally, salting means that it is impossible to determine whether two users have the same password without cracking the hashes, as the different salts will result in different hashes even if the passwords are the same.
- Modern hashing algorithms such as **Argon2id, bcrypt, and PBKDF2** automatically salt the passwords, so no additional steps are required when using them.

# Password Hashing Algorithms

- There are a number of modern hashing algorithms that have been specifically designed for securely storing passwords. This means that they should be slow (unlike crypto hashes such as SHA family & KECCAK, which were designed to be fast), and how slow they are can be configured by changing the [work factor](#).
- [Argon2](#) is the winner of the 2015 [Password Hashing Competition](#).
- The [bcrypt](#) password hashing function should be the second choice for password storage if Argon2 is not available

Is security highest if users are forced to use long passwords, mixing upper and lower case characters and numerical symbols, generated for them by the system, and changed repeatedly?

1. Users may have difficulty memorizing complex passwords.
2. Users may have difficulty dealing with frequent password changes.
3. Users may find ways of re-using their favourite password.



Passwords will be written on a piece of paper kept close to the computer

***Is it always a bad idea  
to write down your  
password?***

# PASSWORD POLICIES

1

## Set a password

If there is no password for a user account, the attacker does not even have to guess it.

2

## Change default passwords

Often passwords for system account have a default value like “manager”.

- Default passwords help field engineers installing the system; if left unchanged, it is easy for an attacker to break in.
- Would it then be better to do without default passwords?

3

## Avoid guessable passwords

- Prescribe a minimal password length.
- Password format: mix upper and lower case (case-sensitive), include numerical and other non-alphabetical symbols (alphanumeric).
- Today on-line dictionaries for almost every language exist.

4

## Password ageing

- Set an expiry dates for passwords to force users to change passwords regularly.
- Prevent users from reverting to old passwords, e.g. keep a list of the last “ten” passwords used.

5

## Limit login attempts

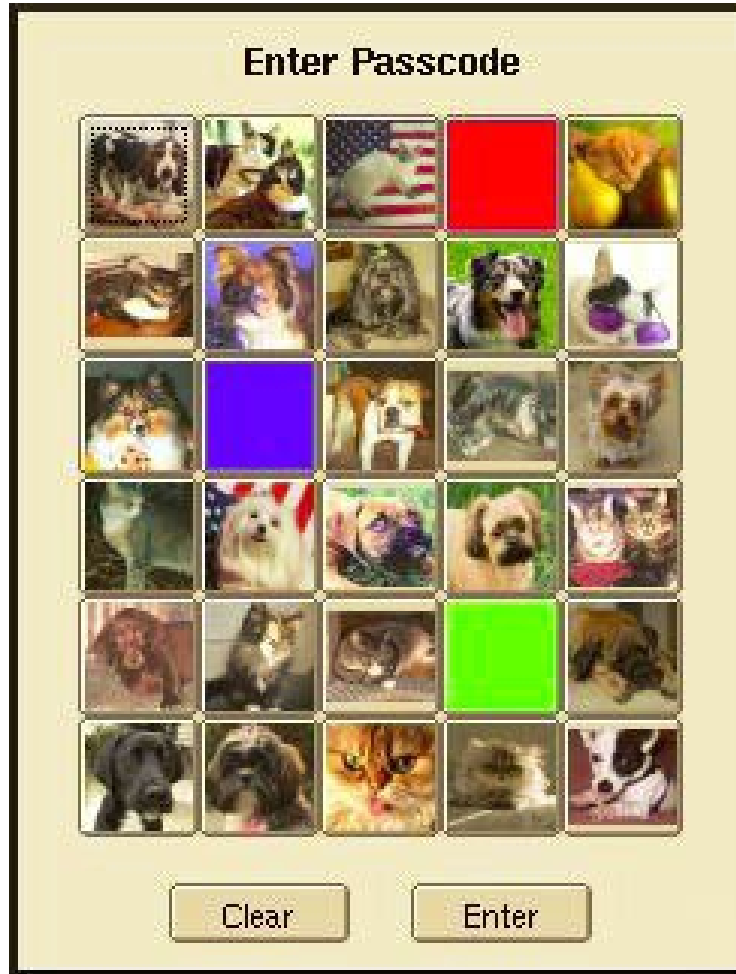
The system can monitor unsuccessful login attempts and react by locking the user account (completely or for a given time interval) to prevent or discourage further attempts.

6

## Inform user

After successful login, display time of last login and the number of failed login attempts since, to warn the user about recently attempted attacks.

# ALTERNATIVE FORMS OF PASSWORD



1

## Passphrase

User enters sentences or long phrases that are easy to remember, and the system applies a hash function to compute the (fixed-size) actual passwords.

2

## Visual drawing patterns

(on touch interface), used in, e.g. Android.

3

## Picture passwords

Select objects in pictures and patterns. Used in Windows 8.

4

## One-time passwords.



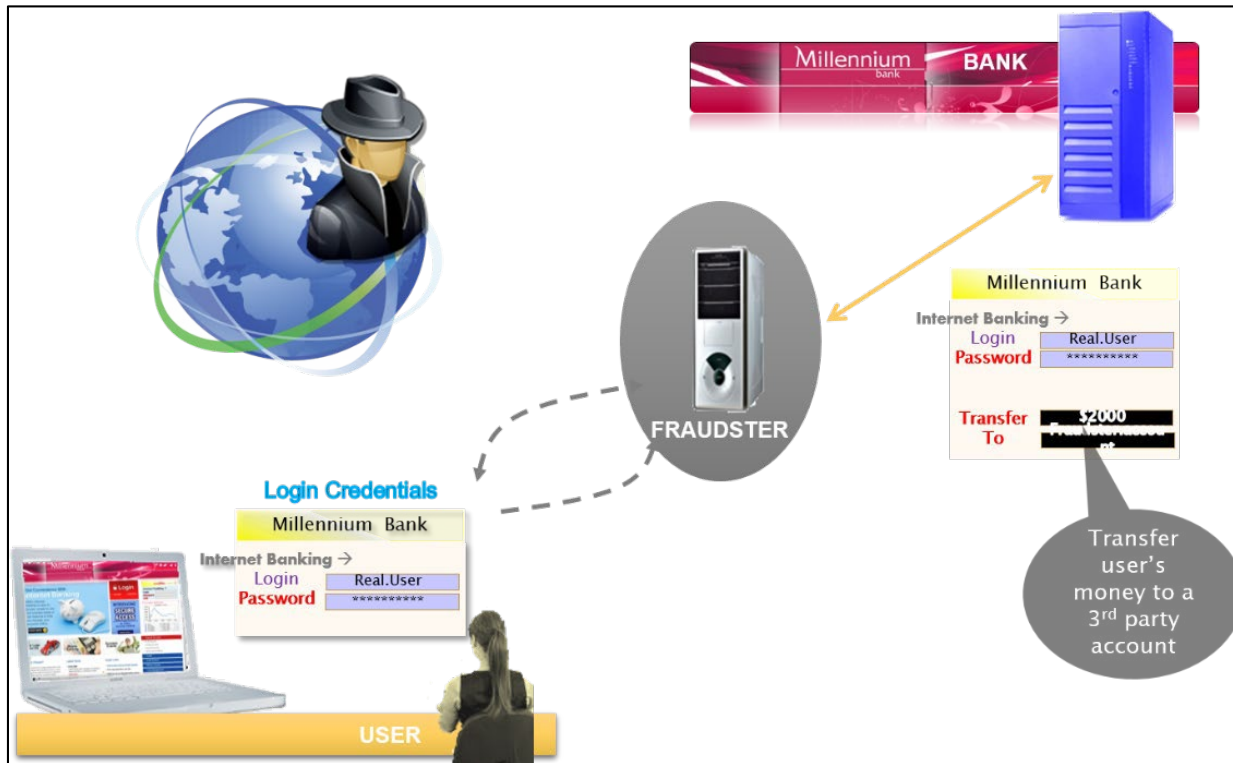
# ONE TIME PASSWORD



- The *one-time passwords* scheme attempts to address a key weakness in the password-based scheme: reuse of stolen passwords.
- The idea is to generate a list of passwords, and each password is *used only once*.
- We'll discuss one scheme based on *Lamport's one-time passwords*.
- Lamport's scheme uses a one-way function, e.g., cryptographic hash, to generate a sequence of passwords.

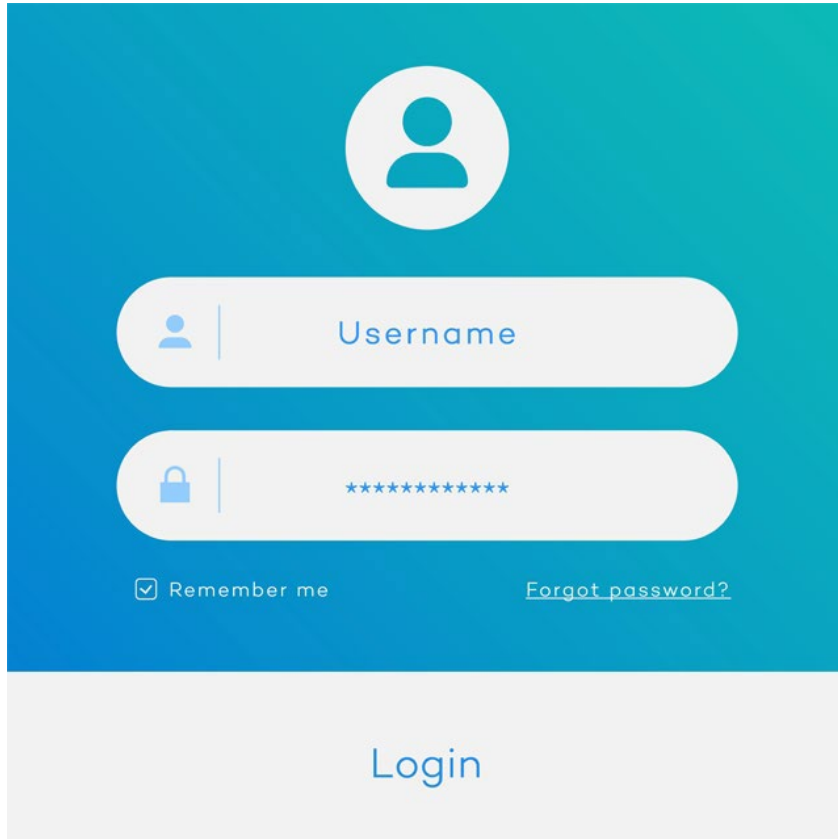
*In practice, most OTP systems in use today are based on the HOTP Algorithm (An HMAC-Based One-Time Password Algorithm) specified in IETF RFC 4226 (<https://tools.ietf.org/html/rfc4226>)*

# PHISHING AND SPOOFING



- Identification and authentication through username and password provide **unilateral authentication**.
- Computer verifies the user's identity but the user has no guarantees about the identity of the party that has received the password.
- In **phishing** and **spoofing** attacks, a party voluntarily sends the password over a channel, but is misled about the end-point of the channel.

# SPOOFING ATTACKS



Username

\*\*\*\*\*

☒ Remember me [Forgot password?](#)

Login

1. Attacker starts a program that presents a fake login screen and leaves the computer.
2. If the next user coming to this machine enters username and password on the fake login screen, these values are captured by the program.
  - Login is then typically aborted with a (fake) error message and the spoofing program terminates.
  - Control returned to operating system, which now prompts the user with a genuine login request.

- **Display number of failed logins**: may indicate to the user that an attack has happened.
- **Trusted path**: guarantee that user communicates with the operating system and not with a spoofing program; e.g., Windows has a **secure attention key** CTRL+ALT+DEL for invoking the operating system logon screen.
- **Mutual authentication**: user authenticated to system, system authenticated to user.

Remember the distinction between Simple Authentication and Strong Authentication.

## Phishing



Attacker impersonates the system to trick a user into releasing the password to the attacker.

- A message could claim to come from a service you are using, tell you about an upgrade of the security procedures, and ask you to enter your username and password at the new security site that will offer stronger protection.

## Social Engineering

Attacker impersonates the user to trick a system operator into releasing the password to the attacker.

**Take care to enter your passwords only at the “right” site (but how do you know?)**

# PROTECTING THE PASSWORD FILE

## Password File



Operating system maintains a file with user names and passwords

Attacker could try to compromise the confidentiality or integrity of this [password file](#).

- Options for protecting the password file:
  - cryptographic protection,
  - access control enforced by the operating system,
  - combination of cryptographic protection and access control, possibly with further measures to slow down dictionary attacks.

1. Only privileged users must have **write access** to the password file.
  - Otherwise, an attacker could get access to the data of other users simply by changing their password, even if it is protected by cryptographic means.
2. If read access is restricted to privileged users, then passwords in theory could be stored unencrypted.
3. If password file contains data required by unprivileged users, passwords must be “encrypted”; such a file can still be used in dictionary attacks.
  - Thus modern Unix/Linux system hides the actual password file in `/etc/shadow` that is not accessible to non-privileged users.

- 1** Measure similarity between reference features and current features.
- 2** User is accepted if match is above a predefined threshold.

NEW ISSUE

## False Positive

Accept wrong user: security problem.

## False Negative

Reject legitimate user: creates embarrassment and an inefficient work environment.



# FORGED FINGERS

## Fingerprints and Biometric Traits

In general, may be unique but they **are no secrets**.

- In September 2013, hackers show how to lift fingerprints from iPhone 5s. Similar attacks also apply to Samsung S5 phone.  
<http://www.ccc.de/en/updates/2013/ccc-breaks-apple-touchid>

## Rubber Fingers

Rubber fingers have defeated many commercial fingerprint recognition systems in the past.

- Minor issue if authentication takes place in the presence of security personnel.
- When authenticating remote users additional precautions have to be taken to counteract this type of fraud.

## Secure Storage

Secure storage of biometric data is an important requirement from the angle of personal privacy protection.

## Suggested further reading on authentication (not mandatory):

- Menezes et al. *Handbook of Applied Cryptography*. Chapter 10. <http://cacr.uwaterloo.ca/hac/>
- R. Anderson. Security Engineering. Chapter 15. <https://www.cl.cam.ac.uk/~rja14/book.html>
- [2<sup>nd</sup> edition free & downloadable](#)