# Password security & Authentication Analysis

## How Passwords Are Storred ( Hashing vs Encryption)

### 1)Hashing (used for passwords):-

- Hashing converts a password into a fixed-length hash using a one-way function.
- Cannot be reversed back to the original password.
- When you log in, the system hashes the entered password and compares it with the stored hash.
- Even if the database is stolen, attackers can't directly read passwords.

Best practices

- Add a salt (random data) to each password before hashing.
- Use slow, adaptive algorithms (e.g., bcrypt , Argon2).
  Example: Password: MyPass123 → Hash: $2b$10$... (original password can't be recovered)

**2) Encryption (not recommended for passwords):-**

- Encryption converts data into ciphertext using a key.
- It is reversible: with the key, the original password can be decrypted.
- If the key is compromised, all passwords are exposed.

Used for: data that must be recovered later (files, backups), not user passwords.

# Hash Types & Generating Password Hashes

### 1) MD5 (Message Digest 5):-

- Produces a 128-bit hash (32 hex characters).
- Very fast → easy for attackers to crack.
- Broken & insecure (collisions exist).
- Not safe for passwords.

Example: hello → 5d41402abc4b2a76b9719d911017c592

### 2)SHA-1 (Secure Hash Algorithm 1):-

- Produces a 160-bit hash (40 hex characters).
- More secure than MD5, but now cryptographically broken.
- Vulnerable to collision attacks.
- Not recommended for passwords.

Example: hello → aaf4c61ddcc5e8a2dabede0f3b482cd9aea9434d

### 3) bcrypt:-

- Designed specifically for password hashing.
- Uses salt automatically.
- Slow & adaptive (resists brute-force attacks).
- Widely used in real systems.
- Secure for passwords.

Example format: $2b$12$KIX9...

## Generation of password hashes:-

### 1.MD5 hash:-

42f749ade7f9e195bf475f37a44cafcb

- Fast, insecure
- Do not use for passwords

### 2 SHA-1 hash:-

b2e98ad6f6eb8508dd6a14cfa704bad7f05f6fb1

- Stronger than MD5, but broken
- Not recommended

### 3 bcrypt hash:-

$2b$12$wZ0KQF8zE1jXJxv0Hq8K9e5J8H6Ww9YyJrZ3cC1Y7B5mKpZ0a

- Includes salt + cost factor
- Slow and secure
- Best for password storage

## Cracking weak hashes using wordlists

### What is a wordlist attack?

- A wordlist is a file containing common passwords (e.g., password, 123456).
- The attacker hashes each word and compares it with the stolen hash.
- If hashes match → the password is found.

### How cracking works (simple steps):-

- Get a hash (e.g., MD5 or SHA-1)
- Load a wordlist
- Hash each word using the same algorithm
- Compare with the target hash
- Match found → password cracked

Example (MD5)

- Hash: 5d41402abc4b2a76b9719d911017c592
- Wordlist word: hello
- MD5(hello) = 5d41402abc4b2a76b9719d911017c592
- password cracked: hello

### Why weak hashes are easy to crack

- Fast algorithms (MD5, SHA-1) → millions of guesses per second
- No salt → same password = same hash
- Common passwords exist in wordlists

## Brute Force vs Dictionary Attack

### 1)Brute-Force Attack:-

- Tries every possible combination of characters.
- Example: a → b → c → ... → aa → ab → ac
- Guaranteed to work if given enough time.
- Very slow for long or complex passwords.

**Best against:-**

- Short passwords
- No rate-limiting

### 2) Dictionary Attack:-

- Tries common passwords from a wordlist.
- Example: password, admin123, qwerty
- Much faster than brute force.
- Fails if password is uncommon or random.

**Best against:-**

- Human-chosen passwords
- Reused or leaked passwords

## Why Weak Passwords Fail

### 1)Too short:-

- Short passwords have fewer combinations.
- Example: 1234 can be cracked in seconds.

### 2) Common & predictable:-

- Passwords like password, admin, qwerty exist in wordlists.
- Dictionary attacks crack them almost instantly.

## 3) No complexity:-

- Using only lowercase letters or numbers makes guessing easy.
- Example: abcdef, 111111

## 4) Reused passwords:-

- One breach → attackers try the same password everywhere (credential stuffing).

## 5) Weak hashing on servers:-

- Stored with MD5 or SHA-1 (fast, unsalted).
- Attackers can crack millions per second.

## 6) No rate limiting:-

- Unlimited login attempts allow brute-force attacks.

**In simple words:-**

- Weak passwords = easy guesses + fast cracking
- Attackers don't "hack" — they guess very fast

# What is MFA & Its Importance

**MFA (Multi-Factor Authentication) means proving your identity using two or more different factors instead of just a password.**

## Authentication factors (types):-

### 1) Something you know:-

- Password, PIN

### 2)Something you have:-

- Phone, OTP app, SMS code, security key

### 3)Something you are:-

- Fingerprint, face, iris

**MFA = any two or more of these together**

**Example**

- Password + OTP on phone
- Password + fingerprint
- PIN + security key

**Even if a password is stolen, login still fails.**

## Why MFA is important:-

- Stops password attacks (brute force, dictionary, phishing)
- Protects against stolen credentials
- Reduces account takeover risk drastically
- Used by banks, email, social media, cloud services

# Recommendations For Strong Authentication

## 1) Use Multi-Factor Authentication (MFA):-

- Combine password + OTP / authenticator app / biometrics.
- Protects accounts even if passwords are stolen.
- Prefer app-based MFA over SMS.

## 2) Create strong passwords:-

- Use 12–16+ characters.
- Mix letters, numbers, and symbols.
- Avoid names, dates, or common words.
- Use unique passwords for each service.

## 3) Use password managers:-

- Generate and store strong, random passwords.
- Prevents password reuse.
- Examples: browser password managers, Bitwarden.

## 4) Secure password storage (server-side):-

- Store passwords using bcrypt or Argon2.
- Always use salt.
- Avoid MD5 and SHA-1.

## 5) Limit login attempts:-

- Enable rate limiting and account lockout.
- Stops brute-force attacks.

## 6)Use modern authentication methods:-

- Enable biometrics where available.
- Consider hardware security keys for critical accounts.

## 7) Monitor & educate users:-

- Detect unusual login behavior.
- Teach users to identify phishing attempts.

## In simple words

**Strong password + MFA + secure storage = strong authentication .**