# 🔐 Password Hashing Techniques

Password hashing is a crucial technique for securely storing passwords in databases. Instead of saving plain-text passwords, which can be stolen if the database is compromised, hashing ensures that even if the data is leaked, recovering the original password is extremely difficult.

## 🔍 Commonly Used Password Hashing Techniques

### 1. MD5 (Message Digest Algorithm 5)

• Output Length: 128-bit (32 hexadecimal characters)  
• Security: ❌ Weak (due to collision vulnerabilities)  
• Usage: Previously used for password hashing but now considered insecure due to fast brute-force attacks.  
• Example: md5("password") → 5f4dcc3b5aa765d61d8327deb882cf99

### 2. SHA Family (Secure Hash Algorithm)

• Variants: SHA-1, SHA-256, SHA-512  
• Security:  
 - SHA-1: ❌ Deprecated due to collision attacks.  
 - SHA-256 & SHA-512: ⚠️ Stronger, but not ideal for password hashing due to speed.  
• Example (SHA-256): sha256("password") → 5e884898da28047151d0e56f8dc62927...

### 3. bcrypt

• Key Features: Slow, adaptive hashing with salt.  
• Security: ✅ Strong, designed for password hashing.  
• Salting: Automatically adds salt to prevent rainbow table attacks.  
• Work Factor: Adjustable (cost factor determines computational complexity).  
• Example: $2a$12$Nw0HQpo.Dl4XJ0N3pyktqu98UJOnvPtHpLPYjZ1BSuD84h6N7V8O6

### 4. PBKDF2 (Password-Based Key Derivation Function 2)

• Key Features: Uses salt and multiple iterations to slow down brute-force attacks.  
• Security: ✅ Strong, but can be optimized with modern hardware.  
• Adjustability: Number of iterations can be increased over time.  
• Example: PBKDF2-HMAC-SHA256(password, salt, iterations)

### 5. scrypt

• Key Features: Memory-hard function, designed to resist GPU/ASIC cracking.  
• Security: ✅ Stronger than PBKDF2 for password hashing.  
• Example: scrypt(password, salt, N, r, p)

### 6. Argon2 (Winner of the Password Hashing Competition)

• Variants:  
 - Argon2d: Resistant to GPU attacks.  
 - Argon2i: Resistant to side-channel attacks.  
 - Argon2id: 🔥 Hybrid of both (recommended).  
• Key Features: Highly secure, memory-hard, and resistant to brute-force attacks.  
• Example: argon2id(password, salt, iterations, memory\_cost, parallelism)

## 🔥 Which Hashing Technique Should You Use?

### ✅ Best Options (Modern & Secure)

• Argon2id (🏆 Recommended)  
• bcrypt  
• scrypt

### ❌ Avoid Using

• MD5, SHA-1, SHA-256 alone (Too fast for password storage, prone to brute-force attacks)

🔗 For best security practices, always use a slow, memory-hard hashing algorithm like Argon2id, bcrypt, or scrypt!