

1. Introduction

Passwords remain the most widely used authentication mechanism across websites, enterprise systems, devices, and cloud platforms.

Despite advancements in authentication, 81% of data breaches occur due to weak or stolen passwords (Verizon DBIR).

This report deeply explores:

how passwords are stored

types of hashing algorithms

how attackers crack passwords

hands-on cracking using John the Ripper

importance of multi-factor authentication

strategies to create secure authentication systems

This is a full practical + theoretical analysis.

2. Password Storage Mechanisms (EXPLAINED IN DEPTH)

2.1 Plain Text Password Storage

Plain text means storing passwords directly in the database like:

username: admin

password: admin123

If database gets leaked → attacker immediately gets all logins.

This is never used in any secure system.

2.2 Hashing (One-Way Transformation)

Hashing converts passwords into unreadable strings.

Example:

Input: password

MD5: 5f4dcc3b5aa765d61d8327deb882cf99

SHA-1: 5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8

Characteristics of Hashing

- ✓ One-way

- ✓ Deterministic
- ✓ Fixed-length output
- ✓ Fast or slow based on algorithm

Hashing ensures passwords remain protected even if the database leaks.

2.3 What Is Salt? (Very Important for Interviews)

Salt

A random string added to a password before hashing:

password + random_salt → hash

Purpose of Salt

- ✓ Makes identical passwords produce different hashes
- ✓ Prevents rainbow table attacks
- ✓ Increases complexity for attackers

Example:

User1: password + XRT12\$ → hashaabb113

User2: password + YHN2!@ → hash91baaccd

2.4 Pepper (Advanced Security Concept)

A server-side secret key applied before hashing.

Unlike salt, pepper is not stored with the password.

 3. Hashing Algorithms Explained (In-Depth)

3.1 Fast Hashes (Weak Today)

MD5

1992 algorithm

Extremely fast → ideal for attackers

Trillions of hashes per second with GPU

Collisions possible

Status: Broken & insecure

SHA-1

More secure than MD5

But also vulnerable

Collision attacks demonstrated by Google in 2017

Status: Deprecated

3.2 Slow Hashes (Secure)

bcrypt

Uses salt automatically

“Cost factor” makes hash generation slow

Slowness = protection against brute force

argon2 (Best & Modern)

Winner of Password Hashing Competition

Resistant to GPU cracking

Recommended for modern systems

4. Hash Type Identification

Before cracking, attacker must identify the hash type.

Tools used:

Hashes.com Hash Identifier

Hashcat — --identify

John the Ripper — --list=formats

Identification Process

Example hash:

5f4dcc3b5aa765d61d8327deb882cf99

Tool identifies it as:

MD5

32 hex characters

Known pattern

Correct identification is necessary for successful cracking.

5. Hands-On Practical – Generate & Crack Passwords

5.1 Environment Used

Kali Linux (main)

Wordlist: rockyou.txt

Tool: John the Ripper

6. Generating Password Hashes (STEP BY STEP)

6.1 Generate MD5 Hash

Command:

```
echo -n "Password123" | md5sum
```

Output:

```
482c811da5d5b4bc6d497ffa98491e38
```

6.2 Generate SHA-1 Hash

```
echo -n "Password123" | sha1sum
```

6.3 Generate bcrypt Hash

```
mkpasswd --method=bcrypt
```

7. Password Cracking Demonstration (MOST IMPORTANT)

7.1 Create hash file

```
echo "5f4dcc3b5aa765d61d8327deb882cf99" > hash.txt
```

7.2 Dictionary Attack (Fastest & Most Common)

Command:

```
john hash.txt --wordlist=/usr/share/wordlists/rockyou.txt
```

Why dictionary attacks work?

Most users use predictable passwords

Leaked lists contain millions of passwords

Attack takes seconds

Expected result:

password

⌚ 7.3 Brute Force Attack (Slow but Guaranteed)

Command:

```
john hash.txt --incremental
```

Brute force tries every possible combination

Examples:

a, b, c, aa, ab, ac, aaa...

Time complexity example

4-letter password → seconds

8-letter password → days

12-letter password → years

🧠 8. Advanced Attacks (Deep Explanation)

8.1 Rainbow Table Attack

Precomputed tables of:

password → hash

Used to instantly reverse weak hashes.

Cannot break:

- ✓ salted hashes
- ✓ bcrypt
- ✓ argon2

8.2 Hybrid Attack

Dictionary + brute force

Examples:

```
password1  
admin123  
user2024!
```

8.3 Mask Attack

Used when pattern is known.

Example:

Format: Name + 123

john hash.txt --mask=?l?l?l?d?d?d

9. Why Weak Passwords Fail (Research-Based)

Based on 2024 cyber security research:

123456 → cracked in 1 second

password → cracked instantly

8-character lowercase only → cracked in minutes

Phone numbers → predictable

Date of birth → widely used

10. Multi-Factor Authentication (MFA) – Detailed Analysis

10.1 Authentication Factors

- ① Something you know (password)
- ② Something you have (OTP, phone)
- ③ Something you are (fingerprint)

10.2 Why MFA Stops 99% Attacks

Even if attacker:

- ✓ cracks password
- ✓ steals password
- ✓ buys password from dark web

They still cannot login without:

OTP

Fingerprint

Authenticator app

10.3 Types of MFA

SMS OTP

Email OTP

Google Authenticator

Microsoft Authenticator

Hardware keys (YubiKey)

Biometric authentication

11. Best Practices for Strong Password Security

11.1 For Users

- ✓ 12+ characters
- ✓ Mix of symbols/numbers
- ✓ No personal info
- ✓ Use password managers
- ✓ Enable MFA everywhere

11.2 For Organizations

- ✓ Use bcrypt/argon2
- ✓ Enable MFA
- ✓ Enforce password rotation
- ✓ Monitor login anomalies
- ✓ Block common passwords
- ✓ Rate-limit login attempts

12. Interview Questions (Expanded Answers)

Q1. What is hashing?

Hashing is a cryptographic transformation converting data into a fixed-length irreversible output.

Q2. Difference between hashing and encryption?

Hashing Encryption

One-way Two-way

Cannot reverse Can reverse with key

Used for passwords Used for data storage & communication

Q3. What is brute force attack?

Trying all possible combinations until correct password is found.

Q4. Importance of MFA?

Adds second layer which prevents unauthorized access even with password.

Q5. What makes a strong password?

Length, complexity, uniqueness, unpredictability.