Hashing, Integrity, Passwords & Cracking

“If you can’t verify it, don’t trust it.”

CЯ4CK1NG-THΞ-C0D3

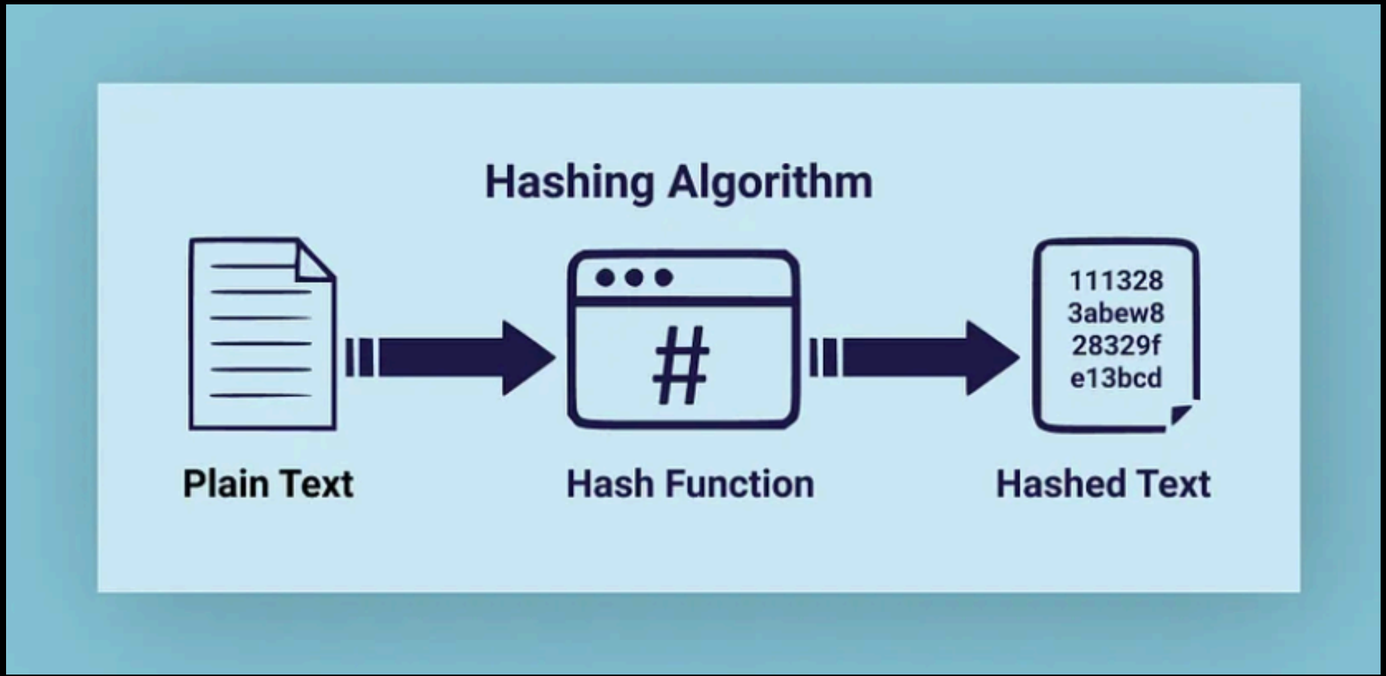
Today’s Roadmap

- Hashing: what & why
- Integrity checks & HMAC
- Secure password storage (salts, pepper, KDFs)
- Cracking techniques & tools (Hashcat / John)

Firstly HASHING & INTEGRITY

What is a Hash?

- A one-way transformation: data → fixed-size digest.
- Deterministic: same input → same output.
- Not reversible.



How are they useful?

Key Properties of Cryptographic Hashes

- **Input Processing:** Accepts inputs of any length (text, file, or data stream).
- **Fixed-Size Output:** Produces a hash of fixed length, regardless of input size.
- **Deterministic:** The same input always results in the same hash value.
- **Avalanche Effect:** A small change in input causes a drastically different hash.
- **One-Way Function:** It’s computationally impossible to reverse the hash to get the original input.
- **Collision Resistance:** Minimizes the chance of two different inputs generating the same hash.

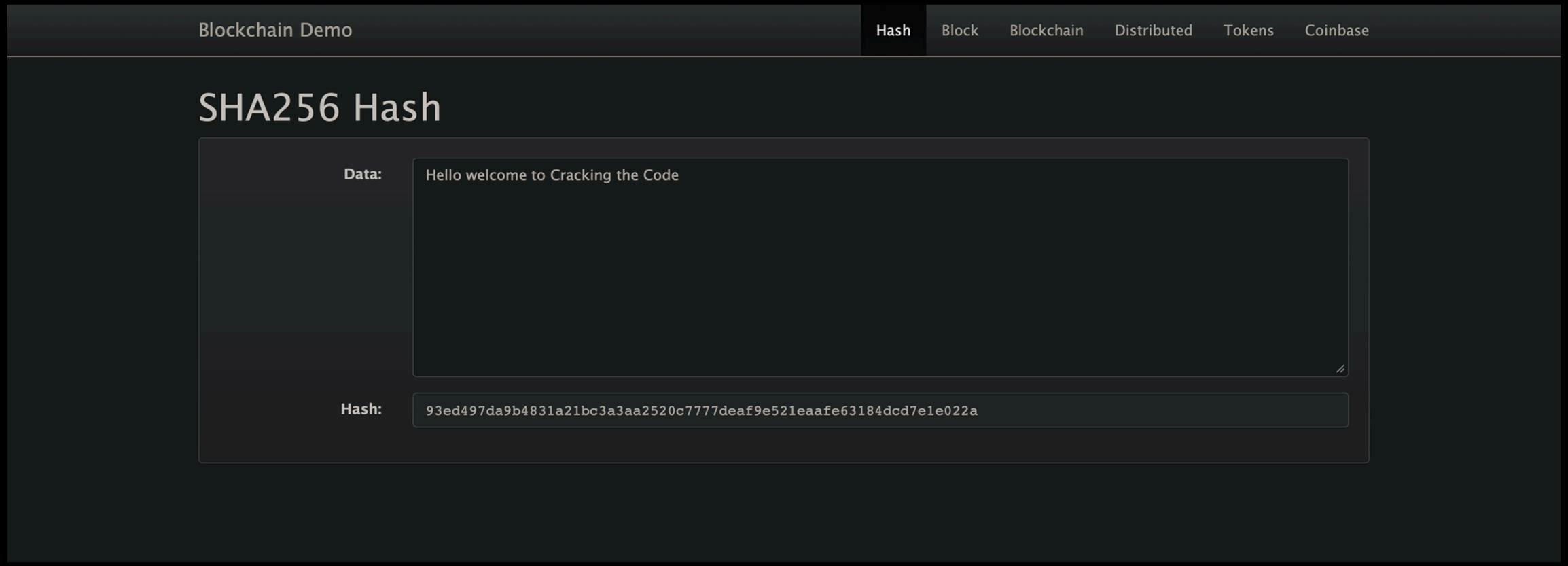
Common Cryptographic Hash Algorithms:

- MD5: Once widely used for checksums and file verification, MD5 is now considered insecure due to discovered collisions, though it remains fast.
- SHA-1: Improved over MD5 but now deprecated for most security applications since it’s also vulnerable to collision attacks.
- SHA-256 / SHA-3: Modern, secure standards used today in digital signatures, blockchain, and data integrity verification.

Visual (Example Output Sizes):

Algorithm	Output Length	Security Status
MD5	128 bits (32 hex)	Broken (collisions)
SHA-1	160 bits (40 hex)	Deprecated
SHA-256	256 bits (64 hex)	Secure
SHA-3	256 bits (64 hex)	Secure (new standard)

Lets try out hashing - SHA256 as a demo



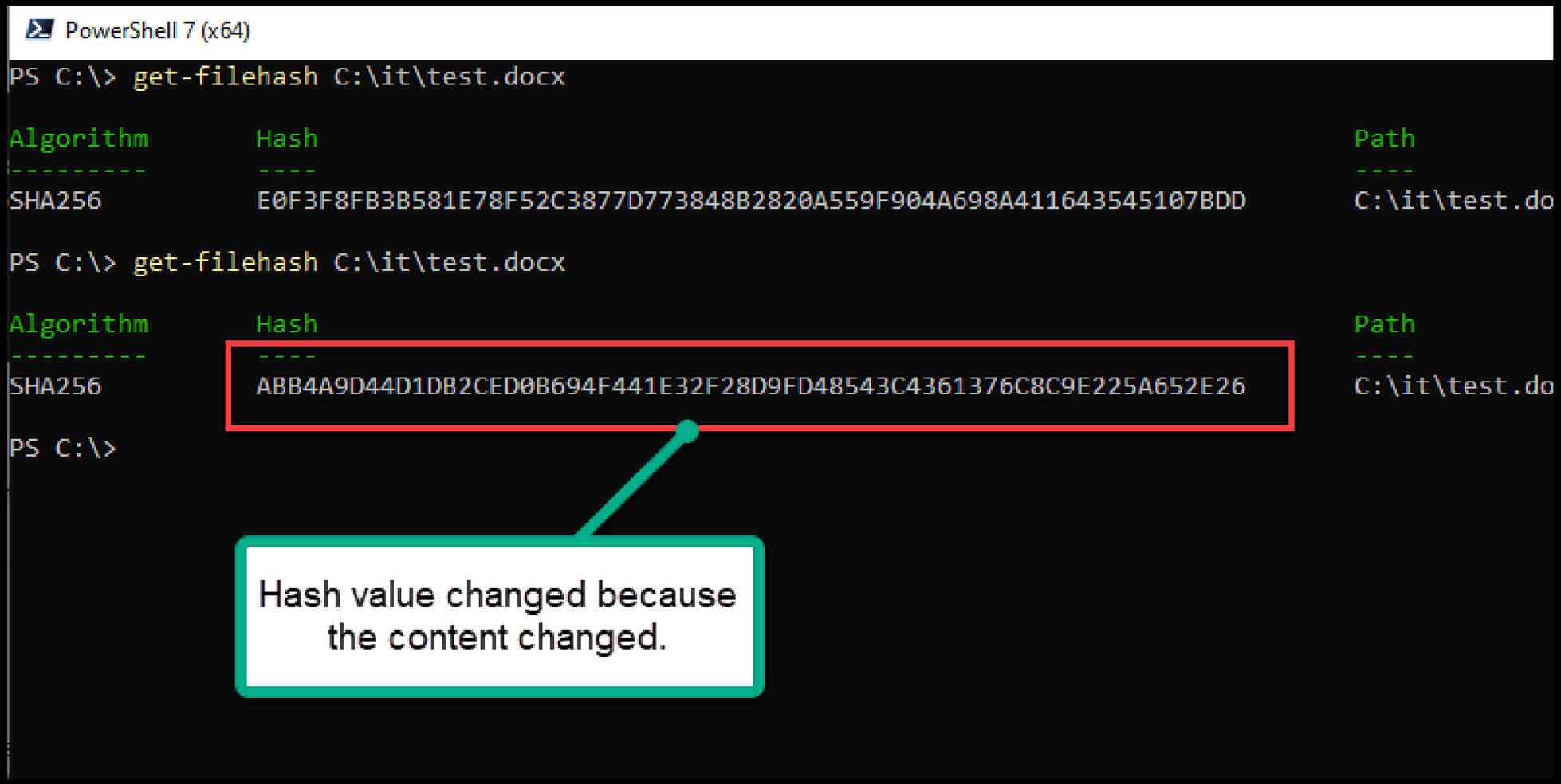
https://andersbrownworth.com/blockchain/hash

Integrity: Why Hashes Matter

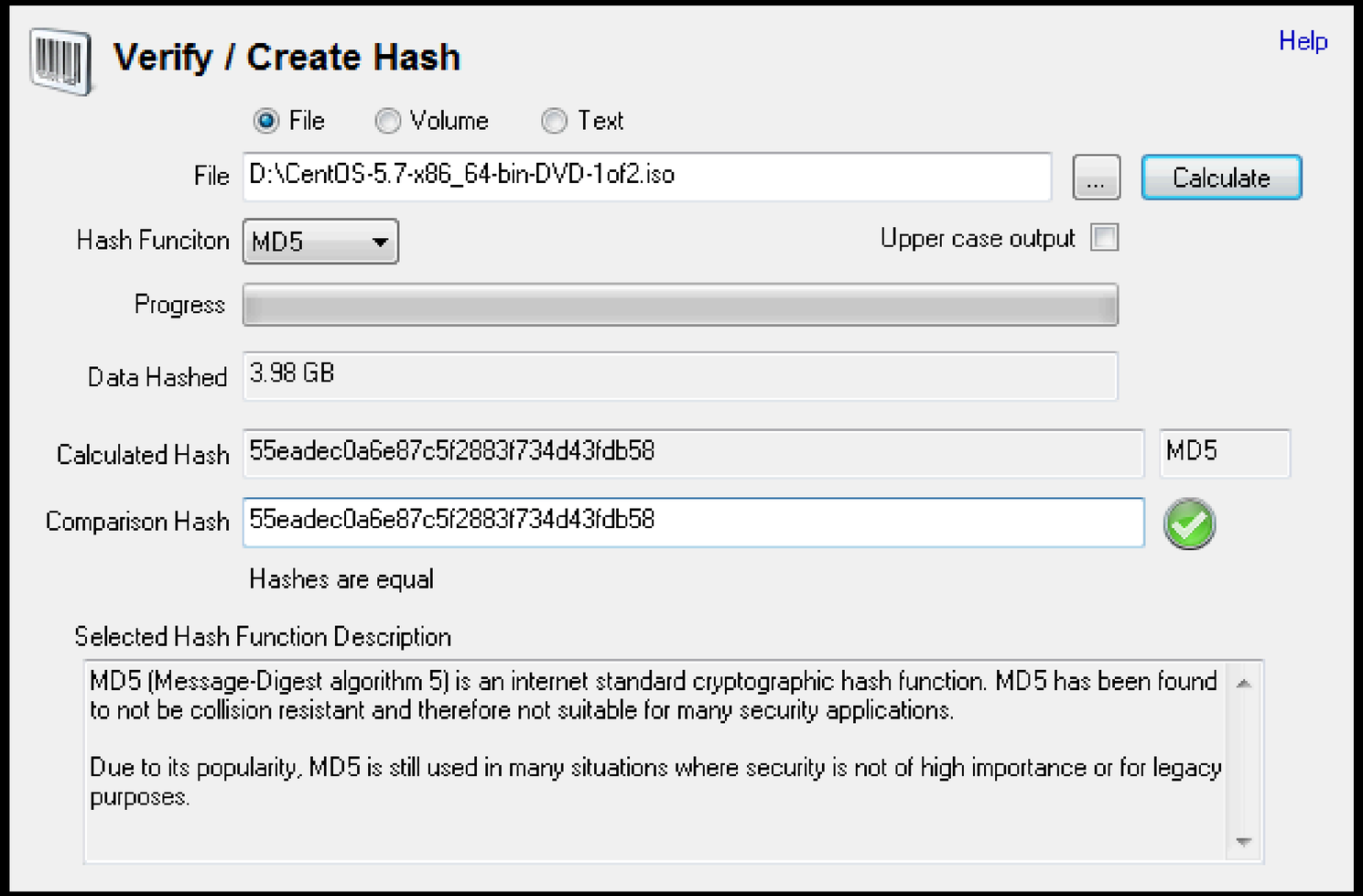
- **Verification of Downloads:** Hashes ensure that files downloaded from the internet are authentic and haven’t been altered.
- **Tamper Detection:** Any modification—intentional or accidental—changes the hash completely, making it easy to detect corruption or malicious interference.
- **Use in Software Systems:** Hashes are widely used in package managers, software updates, and backup systemsto verify that data remains intact and unmodified during transfer or storage.
- **Result Interpretation:** If the computed hash matches the expected one, the file is trusted (green); if it differs, the file has been altered (red).

Lets see some examples

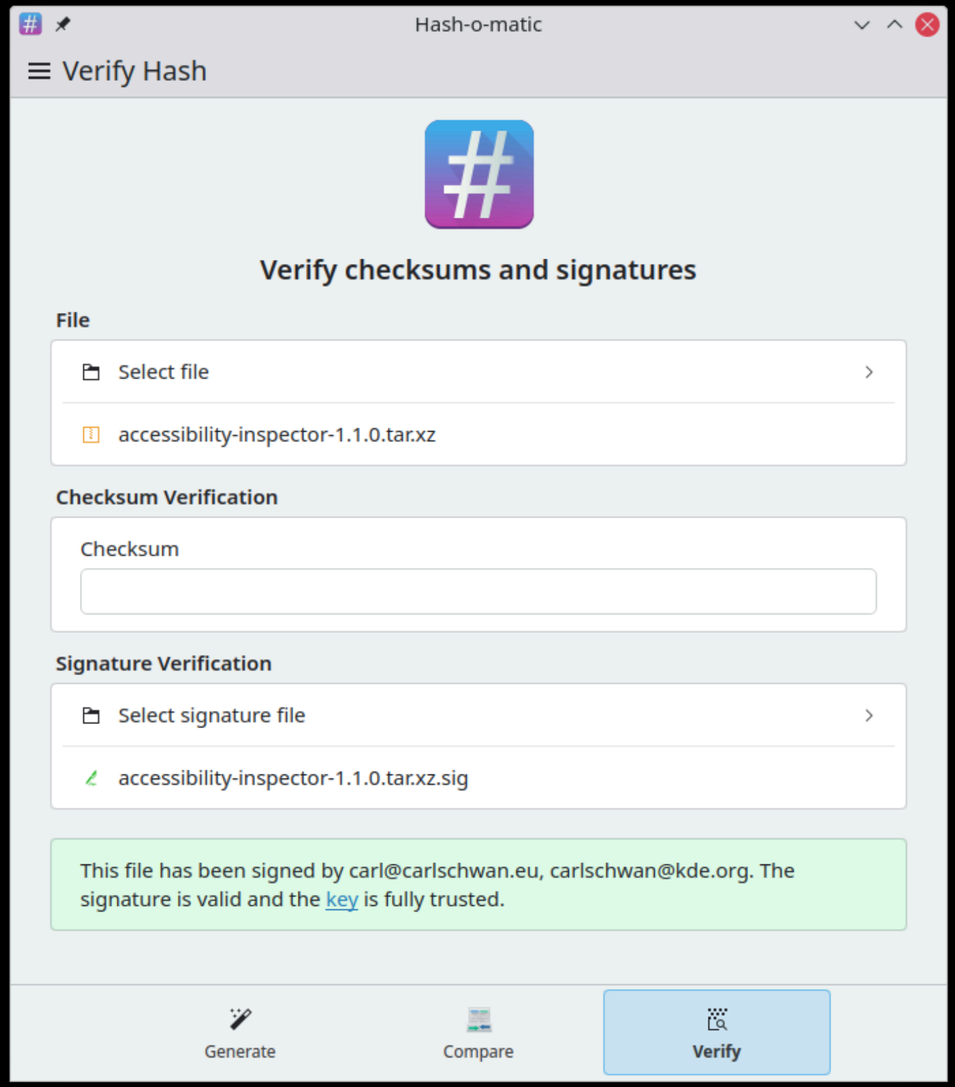
File hashes


PowerShell 7 (x64)
PS C:\> get-filehash C:\it\test.docx
Algorithm Hash Path
SHA256 E0F3F8FB3B581E78F52C3877D773848B2820A559F904A698A411643545107BDD C:\it\test.do
PS C:\> get-filehash C:\it\test.docx
Algorithm Hash Path
SHA256 ABB4A9D44D1DB2CED0B694F441E32F28D9FD48543C4361376C8C9E225A652E26 C:\it\test.do
PS C:\>
Hash value changed because the content changed.

File hashes


Verify / Create Hash
Help
File
Volume
Text
File
D:\CentOS-5.7-x86_64-bin-DVD-1of2.iso
...
Calculate
Hash Funciton
MD5
Upper case output
Progress
Data Hashed
3.98 GB
Calculated Hash
55eadec0a6e87c5f2883f734d43fdb58
MD5
Comparison Hash
55eadec0a6e87c5f2883f734d43fdb58
Hashes are equal
Selected Hash Function Description
MD5 (Message-Digest algorithm 5) is an internet standard cryptographic hash function. MD5 has been found to not be collision resistant and therefore not suitable for many security applications.
Due to its popularity, MD5 is still used in many situations where security is not of high importance or for legacy purposes.

File hashes


Hash-o-matic
Verify Hash
Verify checksums and signatures
File
Select file
accessibility-inspector-1.1.0.tar.xz
Checksum Verification
Checksum
Signature Verification
Select signature file
accessibility-inspector-1.1.0.tar.xz.sig
This file has been signed by carl@carlschwan.eu, carlschwan@kde.org. The signature is valid and the key is fully trusted.
Generate
Compare
Verify

Try it your self in these websites

MD5File.com – File Hash Online Calculator

- This lets you drop or select a file in your browser and computes MD5, SHA-1, SHA-256, SHA-512 locally (without uploading).

Hash-File.online

- Another browser-based calculator for popular hash algorithms.

Toolsley – Hash & Validate

- You can compute or validate hash values (MD5, SHA-1, SHA-256) by uploading a file.

Defuse Security Checksums

- Allows hashing a file or text, supports multiple algorithms, and claims privacy (doesn’t store your data).

Lets try it on your system

Here’s a simple example using Linux/macOS/Windows commands:

On Linux / macOS:

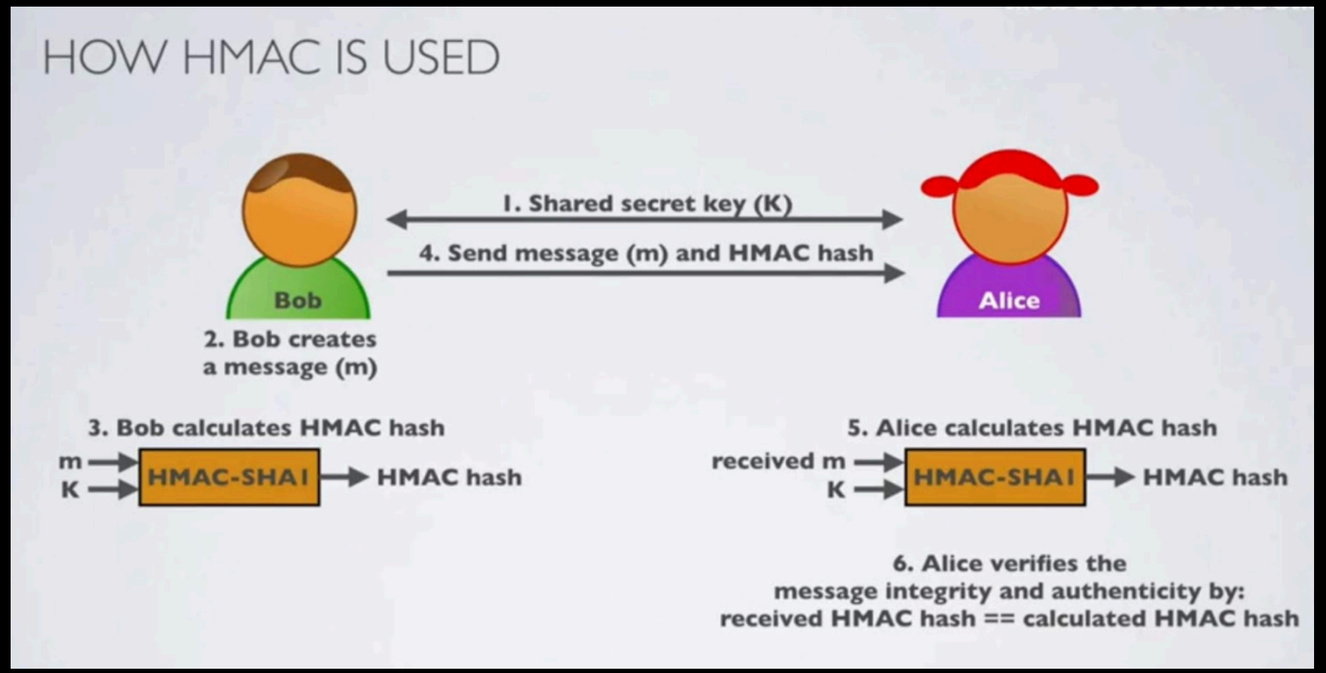
$ sha256sum <myfile.zip>

On Windows (PowerShell)

$ Get-FileHash C:\path\to\file.zip -Algorithm SHA256

HMAC: Authenticated Hashing

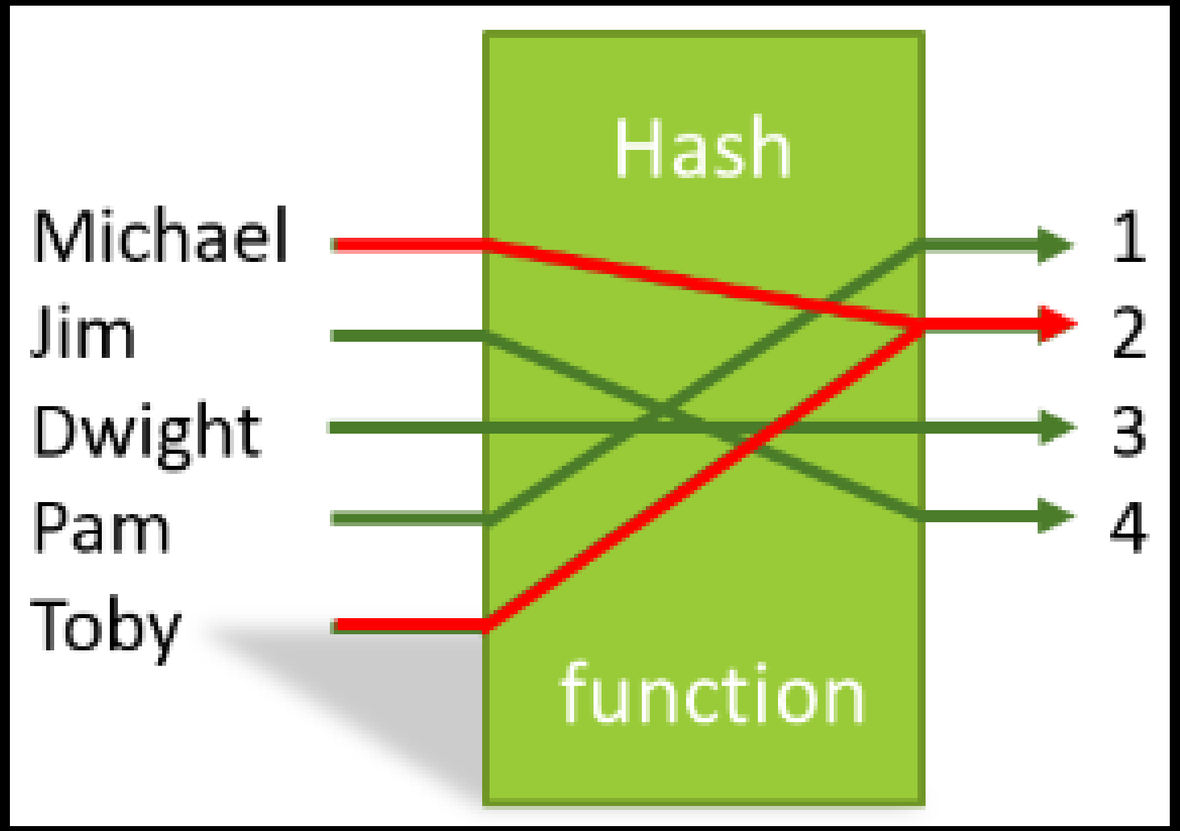
- HMAC = Hash + Secret key → integrity + authenticity.
- Requires shared secret; prevents tampering by third parties.
- Common: HMAC-SHA256 for API authentication.



Lets demo HMAC in Python

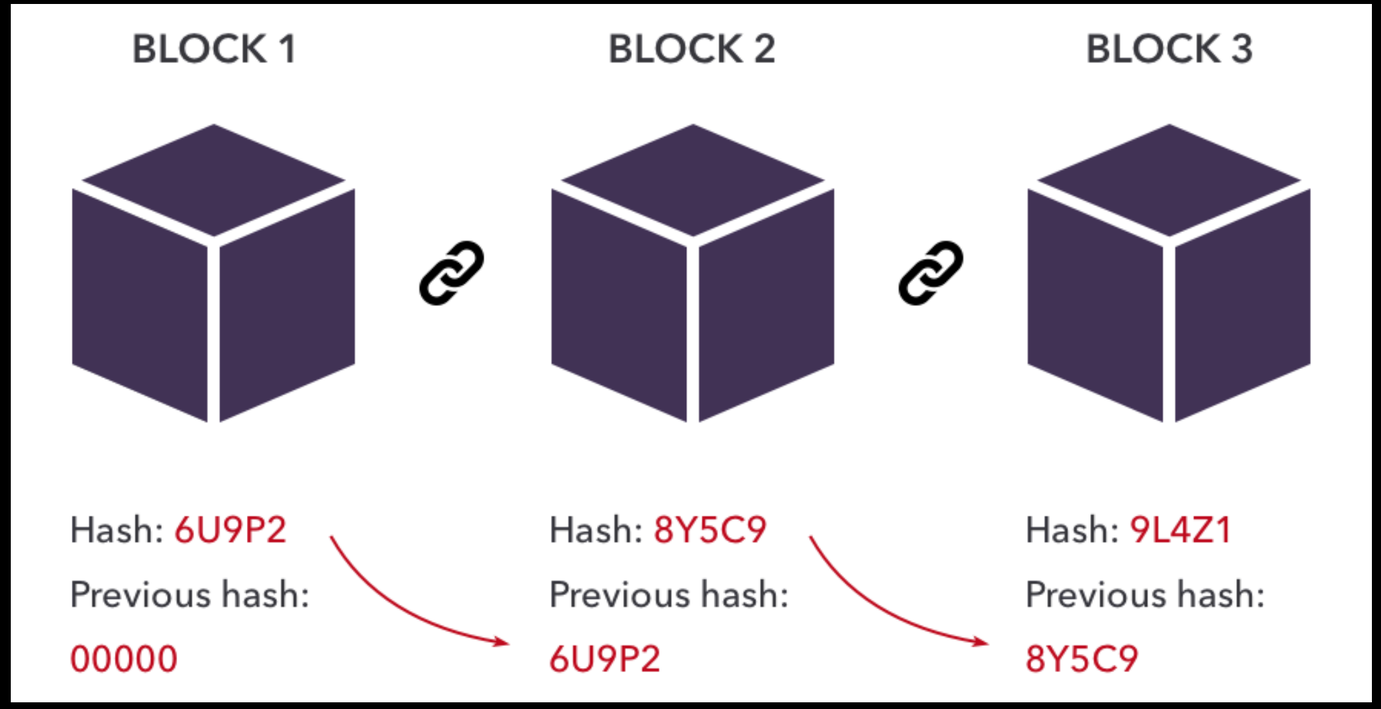
Hashing Pitfalls & Collisions

- Weak hashes (MD5) allow collision attacks (two different files same hash).
- Collisions break trust (e.g., signed packages).
- Always choose modern hashes (SHA-256+).



Hashes in the Real World

- Software package verification (apt, npm).
- Blockchain uses hashing for blocks.
- Password storage (but with extra precautions).



“Hashes help verify data — but for passwords we need extra care. Let’s see why.”

Now lets get to PASSWORDS & CRACKING

Watch video on YouTube

Error 153

Video player configuration error

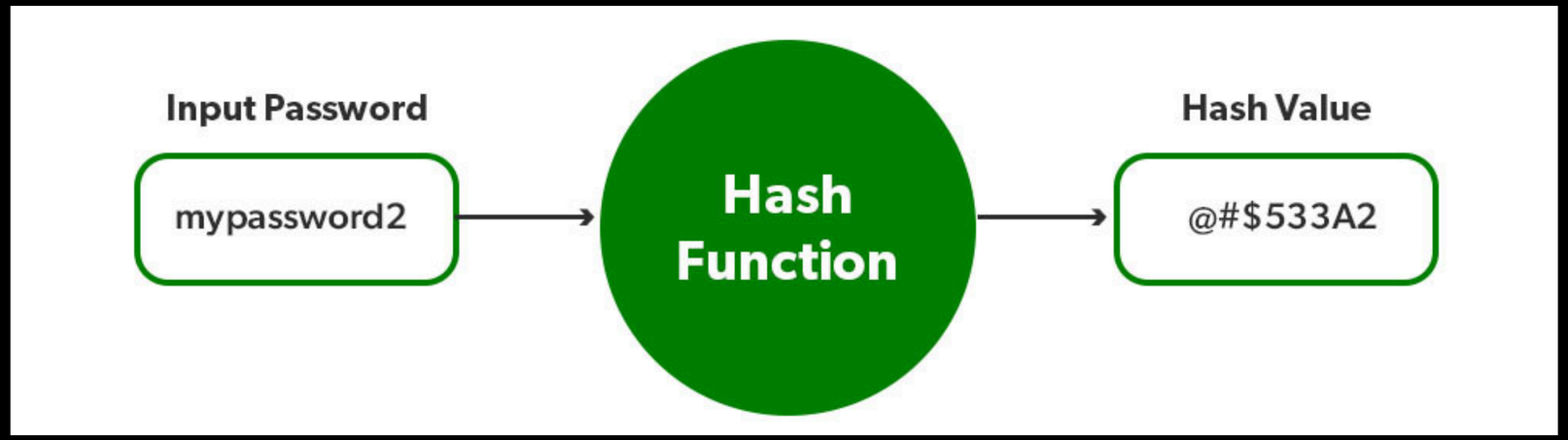
Bad Password Habits

- Reuse across sites.
- Short, predictable strings.
- Storing in plaintext or weak hash.

How is hashing relevant to passwords?

Password Hashing: Not Just Any Hash

- Fast hashes (SHA256) are bad for password storage — they’re too quick to brute-force.
- Use password hashing algorithms: bcrypt, scrypt, Argon2.
- These are slow, tunable, and memory-hard.

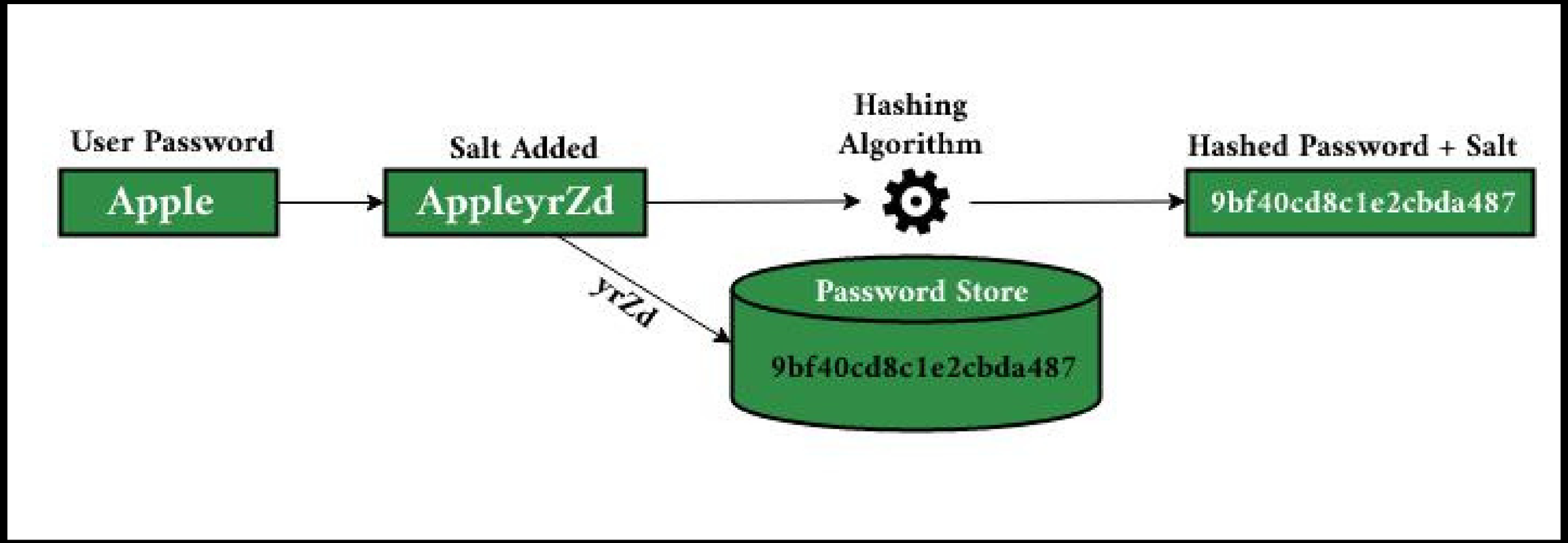


Lets talk about salt

Salts: Per-user Randomness

- Salt: unique random value per password.
- Stored with the hash.
- Prevents rainbow table attacks and makes identical passwords look different.

Password salting is a security technique used to safeguard passwords stored in databases. It involves adding a unique, random string of characters, known as a "salt," to each password before it is hashed (converted into a fixed-length string by a hash function). This salt is then stored along with the hashed password. When the password needs to be verified, the salt is retrieved and combined with the entered password, and the hash function is run again to see if the results match the stored hash.


User Password
Apple
Salt Added
AppleyrZd
yrZd
Hashing
Algorithm
Hashed Password + Salt
9bf40cd8c1e2cbda487
Password Store
9bf40cd8c1e2cbda487

Next up - pepper

You're Probably Using Too Much Black Pepper | Bon Appétit

Pepper & Key Derivation

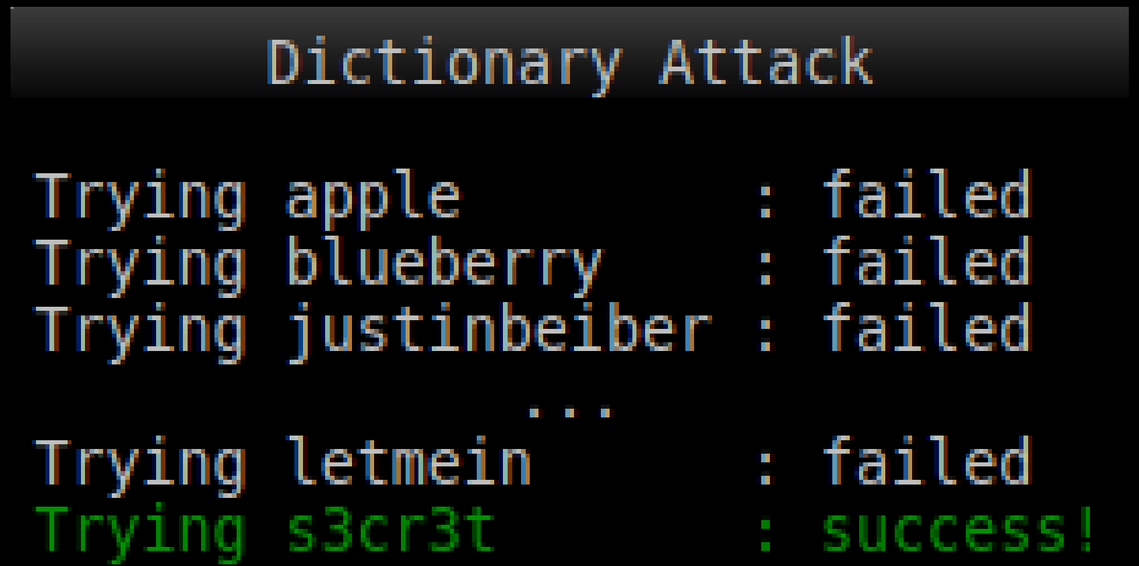
- Pepper: global secret added to hashing (stored separately).
- KDFs: PBKDF2, bcrypt, Argon2 — slow and adjustable cost.
- Use both salt (public) and pepper (secret) for layered defense.

A pepper is a secret added to an input such as a password during hashing with a cryptographic hash function. This value differs from a salt in that it is not stored alongside a password hash, but rather the pepper is kept separate in some other medium, such as a Hardware Security Module. - Wikipedia

Lets think like an attacker now

Attacker Workflow: From Leak to Crack

- Obtain hashed password dump (breach).
- Use wordlists / brute-force / GPU tools.
- Crack weak hashes → hijack accounts.



What are some other type of attacks?

Cracking Techniques

- Dictionary attacks (wordlists)
- Brute-force (all combos)
- Rule-based / mask attacks (patterned guesses)
- Rainbow tables (precomputed) — defeated by salts

Lets get cracking

Tools: Hashcat & John the Ripper

- Hashcat: GPU-accelerated password cracker known for speed and efficiency; supports numerous hash algorithms.
- John the Ripper (JtR): Highly flexible, rule-driven tool; great for custom wordlists and hybrid attacks.
- These tools are powerful and widely used in ethical hacking and digital forensics for password recovery and security testing. Emphasize responsible, authorized use only, never apply them on real systems or data without explicit permission.

Hands on cracking some passwords

Ok so how do we safegaurd against these attacks?

Defenses Against Cracking

- Use long passphrases & password managers.
- Use Argon2/bcrypt with strong params.
- Implement rate-limiting, 2FA, anomaly detection.

Password Policies That Help (and Those That Hurt)

- Encourage length (≥12) and passphrases.
- Avoid forced frequent resets unless breach suspected.
- Don’t require weird complexity rules that push users to predictable variants.

!!Legal & Ethical Reminder!!

- Only test systems you own or have written permission to test.
- Use skills for defense, not misuse.
- Always run tests in isolated lab environments (VMs/staging).
- Use test accounts & synthetic data only — never real user data.
- Document actions, timestamps, and evidence.
- Follow responsible disclosure — report privately and allow fixes.
- Have a rollback plan; inform a contact before testing.
- Unauthorized testing can cause legal or career consequences.