📘

password encryption

💻 Summary of password encryption.

Recall

- What is password encryption?
- What is the process of password encryption?

Overview

Password encryption is essential to store user credentials stored in a database securely. Without password encryption, anyone accessing a user database on a company's servers (including hackers) could easily view any stored passwords.

If someone can read your password on a server, they can use it simply by copy/pasting it—no matter how long or complicated the password might be!

💻 Encryption scrambles your password before saving it on the server. So, if someone hacks the server, instead of finding *password123*, they find a random series of letters and numbers.

To explain password encryption effectively, we must first grasp the language. Several terms might be unfamiliar, so here is a quick intro to password encryption terminology.

- `Key`: Used to lock and unlock passwords using a random string of bits. You get *private and public keys that* crypt and decrypt data differently, but we won't get too deep in the weeds with keys!
- `Bits`: A logical state with one of two possible values, including 1/0, true/false, yes/no, or on/off as typical examples.
- `Block (block cipher)`: A deterministic algorithm operating on fixed-length groups of bits, called blocks.
- `Hash function`: The algorithm that uses the key to create password encryption and decryption. A hash function is essentially a piece of code that runs every time someone saves a password or logs into an application.
- `Hash`: A random series of numbers and letters representing your password. The hash function uses your hash instead of the raw password for authentication.
- `Salt`: Additional letters and numbers appended to the hash

How does it work?

When you save a new password, a hash function creates a hash version and saves that on the server.

Every time you log in using your password, the hash function recreates the hash to see if it matches what's stored. If the hashes match, the algorithm passes the authentication and logs you in.

For example:

- Original password: Pa$$w0rd123
- Hashed password: 6AF1CE202340FE71BDB914AD5357E33A6982A63B

While this might seem secure, simple hashed passwords aren't *hack-proof.*

The hash function only creates a **unique hash for each password**, **not each user**. So, if multiple users have the password, *Pa$$w0rd123*, the hash will be exactly the same.

To overcome this encryption vulnerability, engineers ***salt*** passwords so each hash is unique, even if the passwords are identical.

Salt

A salt appends a unique value of 8 bytes (16 characters) to the password before the hash function creates a hash. This way, even identical passwords are unique before the hash function.

For example:

- Two identical passwords: Pa$$w0rd123
- Salt value one: E1F53135E559C253
- Salt value two: 84B03D034B409D4E
- Password one before hash: Pa$$w0rd123E1F53135E559C253
- Password two before hash: Pa$$w0rd12384B03D034B409D4E
- Password one hashed value (SHA256): 72AE25495A7981C40622D49F9A52E4F1565C90F048F59027BD9C8C8900D5C3D8
- Password two hashed value (SHA256): B4B6603ABC670967E99C7E7F1389E40CD16E78AD38EB1468EC2AA1E62B8BED3A

Common Encryption Methods

Data Encryption Standard (DES)

While applications no longer use Data Encryption Standard (DES), it's important to mention this password encryption method because of its history and influence on more secure modern standards.

IBM developed DES as a 56-bit encryption technology in the early 1970s. The NSA adopted and improved DES before it was approved worldwide as the encryption standard.

However, since the late 70s, hackers have been able to break DES encrypted passwords. In 1999, ethical hackers managed to break a DES key in under 24 hours.

To make DES more secure, engineers created **Triple DES** and later **Advanced Encryption Standard (AES)**, which we still use today.

Triple DES

Triple DES uses three 56-bit keys (blocks) to create 168-bit encryption (some security experts argue that Triple DES is only 112 bits strong). Although Triple-DES is slowly being phased out, many financial institutions still use it to encrypt ATM PINs.

Advanced Encryption Standard

AES is the new encryption standard—trusted by the United States Government and many other prominent organizations globally. At 128 bits, AES is sufficiently secure, but most organizations prefer heavy-duty 256-bit encryption.

Hackers can only break an AES encrypted password through a brute-force attack—trying password combinations to find the right one.

To counter brute-force attacks, applications will lock an account after a certain number of attempts or use tools like Google's reCAPTURE.

Blowfish

American cryptographer, Bruce Schneier, designed Blowfish in 1993 as an antidote to the weak DES encryption. Blowfish uses a 64-bit block and variable key length of 32 to 448 bits.

Although Blowfish is more robust than its DES predecessor, the 64-bit block is still vulnerable to attacks, most commonly *birthday attacks*—a cryptographic attack that exploits the mathematics behind the algorithm.

To solve Blowfish vulnerabilities, engineers created Twofish in 1998 (128-bit blocks with 256-bit keys) and Threefish in 2008 (256, 516, 1024-bit blocks with 256, 516, 1024-bit keys).

Rivest Shamir Adleman (RSA)

RSA is one of the oldest and widely used to transfer data securely. The encryption works with two keys, two large prime numbers, and an additional auxiliary value.

Due to the complicated encrypting and decrypting process, there is no known method for breaking RSA encryption.

Although widely used for transferring data, RSA is slow and not suitable for password encryption.\

Why Strong Passwords Matter

Password encryption can only prevent criminals from viewing saved credentials stored on a server—but cannot prevent hackers from guessing weak/commonly used passwords. If you reuse the same password for multiple accounts, this also puts you at risk!

Encryption is most effective when users create robust, unique passwords for every account. For example, a random 32-character password with letters, numbers, and special characters hashed and salted is near impossible to guess or decode, even using a computer!

In another scenario, let's assume you have a strong 32-character password, but you use it for every account. If hackers manage to steal that password, they have access to every account using the same credentials! Your strong password is effectively useless.

Encrypting in JS

https://code-boxx.com/simple-javascript-password-encryption-decryption/