PHP – Hashing

The term "hashing" represents a technique of encrypting data (specially a text) to obtain a fixed-length value. PHP library includes a number of functions that can perform hashing on data by applying different hashing algorithms such as md5, SHA2, HMAC etc. The encrypted value obtained is called as the hash of the original key.

Processing of hashing is a one-way process, in the sense, it is not possible to reverse the hash so as to obtain the original key.

Applications of Hashing

The hashing technique is effectively used for the following purposes −

Password Authentication

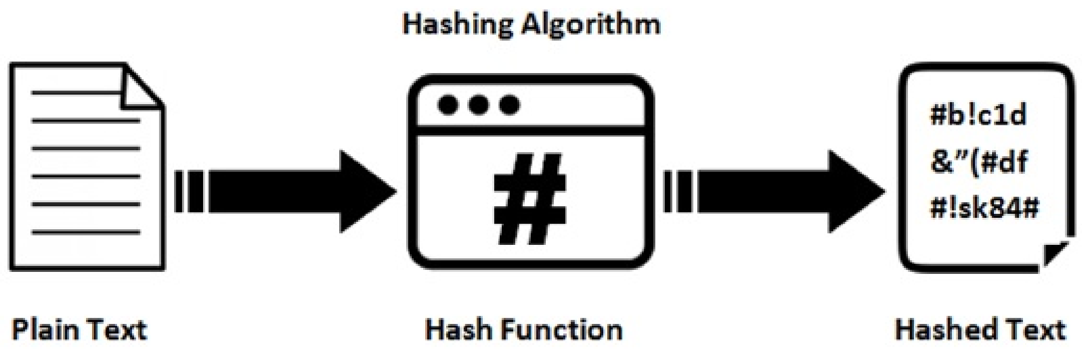
We often register for various online applications such as gmail, Facebook etc. You are required to fill up a form wherein you create a password for an online account. The server hashes your password and the hashed value is stored in the database. At the time of logging in, the password submitted is hashed and compared with the one in the database. This protects your password from being stolen.

Data Integrity

One of the important uses of hashing is to verify if the data has not been tampered with. When a file is downloaded from the internet, you are shown its hash value, which you can compare with the downloaded to make sure that the file has not been corrupted.

The Process of Hashing

The process of hashing can be represented by the following figure −



Hashing Algorithms in PHP

PHP supports a number of hashing algorithms −

- **MD5** − MD5 is a 128-bit hash function that is widely used in software to verify the integrity of transferred files. The 128-bit hash value is typically represented as a 32-digit hexadecimal number. For example, the word "frog" always generates the hash "8b1a9953c4611296a827abf8c47804d7"
- **SHA** − SHA stands for Secure Hash Algorithm. It's a family of standards developed by the National Institute of Standards and Technology (NIST). SHA is a modified version of MD5 and is used for hashing data and certificates. SHA-1 and SHA-2 are two different versions of that algorithm. SHA-1 is a 160-bit hash. SHA-2 is actually a “family” of hashes and comes in a variety of lengths, the most popular being 256-bit.
- **HMAC** − HMAC (Hash-Based Message Authentication Code) is a cryptographic authentication technique that uses a hash function and a secret key.
- **HKDF** − HKDF is a simple Key Derivation Function (KDF) based on the HMAC message authentication code.
- **PBKDF2** − PBKDF2 (Password-Based Key Derivation Function 2) is a hashing algorithm that creates cryptographic keys from passwords.

Hash Functions in PHP

The PHP library includes several hash functions −

The hash_algos Function

This function returns a numerically indexed array containing the list of supported hashing algorithms.

```
hash_algos(): array
```

The hash_file Function

The function returns a string containing the calculated message digest as lowercase hexits.

```
hash_file(
   string $algo,
   string $filename,
   bool $binary = false,
   array $options = []
): string|false
```

The **algo** parameter is the type of selected hashing algorithm (i.e. "md5", "sha256", "haval160,4", etc.). The **filename** is the URL describing location of file to be hashed; supports **fopen** wrappers.

Example

Take a look at the following example −

```
<?php
   /* Create a file to calculate hash of */
   $fp=fopen("Hello.txt", "w");
   $bytes = fputs($fp, "The quick brown fox jumped over the lazy dog.");
   fclose($fp);
   echo hash_file('md5', "Hello.txt");
?>
```

It will produce the following **output** −

```
5c6ffbdd40d9556b73a21e63c3e0e904
```

The hash() Function

The hash() function generates a hash value (message digest) −

```
hash(
   string $algo,
   string $data,
   bool $binary = false,
   array $options = []
): string
```

The **algo** parameter is the type of selected hashing algorithm (i.e. "md5", "sha256", "haval160,4", etc..). The **data** parameter is the message to be hashed. If the binary parameter is "**true**", it outputs raw binary data; "false" outputs lowercase hexits.

Example

The function returns a string containing the calculated message digest as lowercase hexits.

Open Compiler

```
<?php
   echo "Using SHA256 algorithm:" . hash('sha256', 'The quick brown fox jumped ove
   echo "Using MD5 algorithm:",hash('md5', 'The quick brown fox jumped over the la
   echo "Using SHA1 algorithm:" . hash('sha1', 'The quick brown fox jumped over th
?>
```

It will produce the following **output** −

```
Using SHA256 algorithm:68b1282b91de2c054c36629cb8dd447f12f096d3e3c587978dc22
Using MD5 algorithm:5c6ffbdd40d9556b73a21e63c3e0e904
Using SHA1 algorithm:c0854fb9fb03c41cce3802cb0d220529e6eef94e
```