**6 Fungsi Enkripsi di PHP untuk Mengamankan Data**

[**#PHP**](https://www.petanikode.com/topik/php) [**#Enkripsi**](https://www.petanikode.com/topik/enkripsi) [**#Kriptografi**](https://www.petanikode.com/topik/kriptografi) [**#Security**](https://www.petanikode.com/topik/security)



Data adalah harta karun yang paling berharga dalam sebuah sistem. Apabila itu dicuri…

…bencana besar bisa terjadi 😄.

Enkripsi adalah teknik untuk mengamankan data-data tersebut agar isinya tidak diketahui orang lain.

Enkripsi biasanya dilakukan terhadap data-data sensitif seperti password.

Enkripsi akan menjamin data-data tetap aman meskipun berada di tangan orang lain, karena mereka tidak tahu isi aslinya.

Pada kesempatan ini, saya akan membahas beberapa fungsi enkripsi yang sudah disediakan oleh PHP, diantaranya: password\_hash(), crypt(), md5(), hash(), sha1(), dan base64\_encode().

Masing-masing fungsi memiliki algoritmanya tersendiri, ada yang kuat ada juga yang lemah.

Mari kita bahas satu-per-satu…

**1. Fungsi**password\_hash()

Fungsi ini akan menghasilkan sebuah kode *hash* baru dengan metode *one-way hashing*. *one-way hasing* artinya, hasil enkripsinya tidak bisa dikembalikan seperti semula *(decrypt/decode)*.

Contoh:

<?php echo password\_hash("PetaniKode", PASSWORD\_DEFAULT); ?>

Hasilnya:

$2y$10$7PZGQcOAK6ZJXCAO3UFRC.hpv8hjKsQhy5Y7xBOhLNoFNTuxF3lr2

Fungsi ini sangat disarankan untuk mengenkripsi password, karena sulit didekripsi atau di-crack.

Sebelum mengenal fungsi ini, saya biasanya menggunakan fungsi md5(). Namun, fungsi md5() sangat mudah di-crack.

Karena itu, sebaiknya menggunakan fungsi password\_hash() untuk melindungi password.

**Contoh penggunaan**:

Sebelum menyimpan password ke database, sebaiknya dienkripsi terlebih dahulu:

<?php

*$username* = *$\_POST*['username'];

// enkripsi dulu...

*$password* = password\_hash(*$\_POST*['username'], PASSWORD\_DEFAULT);

// query simpan

mysqli\_query("INSERT INTO user (username, password) VALUE ('$username','$password')");

?>

Fungsi passowrd\_hash() ini tidak bisa bekerja sendirian, dia memiliki teman bernama password\_verify().

Fungsi password\_verify(), biasanya digunakaan saat membuat login untuk membandingkan password yang dikirim user dengan password yang tersimpan pada database.

Contoh:

<?php

// ambil data dari form login

*$username* = *$\_POST*['username']; //<-- ini belum aman dari sql injection :D

// ambil data dari database

*$query* = mysqli\_query("SELECT \* FROM user WHERE username='$username'");

*$user* = mysqli\_fetch\_assoc(*$query*);

// bandingkan password yang dikirim dari form login dengan password

// yang ada di database

if( password\_verify(*$\_POST*['password'], *$user*['password']) ) {

// login berhasil

} else {

// login gagal

}

?>

**Tips**: bila ingin menyimpan hasil dari fungsi password\_hash() di database, pastikan bertipe VARCHAR dengan panjang 255 karakter.

Contoh lengkapnya bisa dilihat di: [Tutorial PHP dan MySQL: Membuat Login Register](https://www.petanikode.com/php-login-register)

**2. Fungsi**crypt()

Fungsi ini menghasilkan kode *hash* dengan menggunakan algoritma DES, Blowfish, dan MD5.

Contoh:

<?php echo crypt("petanikode", "garam"); ?>

Hasilnya:

gabeK4a28MgJs

Bila diperhatikan, susuan karakter yang dihasilkan mirip seperti karakter yang digunakan pada alamat URL video youtube.

Mungkin saja youtube juga menggunakan algoritma yang sama dengan fungsi ini.

Pada contoh diatas, kita membarikan dua paramter kepada fungsinya.

Parameter pertama adalah teks yang akan dienkripsi.

Sementara, parameter kedua adalah salt (garam).

[Garam](https://en.wikipedia.org/wiki/Salt_%28cryptography%29) adalah sebuah data acak yang dimasukkan ke dalam fungsi enkripsi.

Fungsi ini juga dapat digunakan untuk mengamankan password, karena sulit di-decode.

**3. Fungsi**md5()

Fungsi ini akan menghasilkan kode *hash* sepanjang 32 karakter.

<?php echo md5("petanikode"); ?>

Hasilnya:

37ebda33c177743169404948552319df

Bila ditambahkan garam, hasilnya akan berubah.

Contoh:

<?php echo md5("petanikode", "g4r4m"); ?>

Hasilnya:

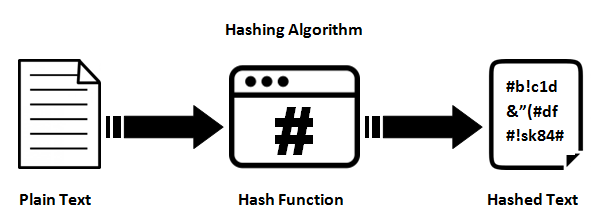
7��3�wt1i@IHU#�

Fungsi ini juga dapat digunakan untuk mengamankan password.

Akan tetapi, kalau kita perhatikan kecepatan komputer zaman sekarang… Sangat mungkin untuk melakukan *decrypt*.

**4. Fungsi**hash()

Fungsi ini akan menciptkan sebuah kode *hash* dengan algoritma tertentu.



Contoh:

<?php echo hash("md5", "petanikode"); ?>

Hasilnya:

37ebda33c177743169404948552319df

Pada contoh di atas, parameter "md5" adalah nama algortima yang akan digunakan untuk mengenkripsi. Parameter "petanikode" adalah teks yang akan dienkripsi.

Bila ingin mengetahui nama-nama algoritma yang bisa digunakan, silahkan coba eksekusi kode ini:

<?php

*$data* = "hello";

foreach (hash\_algos() as *$v*) {

*$r* = hash(*$v*, *$data*, false);

printf("%-12s %3d %s\n", *$v*, strlen(*$r*), *$r*);

}

?>

Hasilnya:

md2 32 a9046c73e00331af68917d3804f70655

md4 32 866437cb7a794bce2b727acc0362ee27

md5 32 5d41402abc4b2a76b9719d911017c592

sha1 40 aaf4c61ddcc5e8a2dabede0f3b482cd9aea9434d

sha256 64 2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e730

sha384 96 59e1748777448c69de6b800d7a33bbfb9ff1b463e44354c3553

sha512 128 9b71d224bd62f3785d96d46ad3ea3d73319bfbc2890caadae2d

ripemd128 32 789d569f08ed7055e94b4289a4195012

ripemd160 40 108f07b8382412612c048d07d13f814118445acd

ripemd256 64 cc1d2594aece0a064b7aed75a57283d9490fd5705ed3d66bf9a

ripemd320 80 eb0cf45114c56a8421fbcb33430fa22e0cd607560a88bbe14ce

whirlpool 128 0a25f55d7308eca6b9567a7ed3bd1b46327f0f1ffdc804dd8bb

tiger128,3 32 a78862336f7ffd2c8a3874f89b1b74f2

tiger160,3 40 a78862336f7ffd2c8a3874f89b1b74f2f27bdbca

tiger192,3 48 a78862336f7ffd2c8a3874f89b1b74f2f27bdbca39660254

tiger128,4 32 1c2a939f230ee5e828f5d0eae5947135

tiger160,4 40 1c2a939f230ee5e828f5d0eae5947135741cd0ae

tiger192,4 48 1c2a939f230ee5e828f5d0eae5947135741cd0aefeeb2adc

snefru 64 7c5f22b1a92d9470efea37ec6ed00b2357a4ce3c41aa6e28e3b

gost 64 a7eb5d08ddf2363f1ea0317a803fcef81d33863c8b2f9f6d7d1

adler32 8 062c0215

crc32 8 3d653119

crc32b 8 3610a686

haval128,3 32 85c3e4fac0ba4d85519978fdc3d1d9be

haval160,3 40 0e53b29ad41cea507a343cdd8b62106864f6b3fe

haval192,3 48 bfaf81218bbb8ee51b600f5088c4b8601558ff56e2de1c4f

haval224,3 56 92d0e3354be5d525616f217660e0f860b5d472a9cb99d6766be

haval256,3 64 26718e4fb05595cb8703a672a8ae91eea071cac5e7426173d4c

haval128,4 32 fe10754e0b31d69d4ece9c7a46e044e5

haval160,4 40 b9afd44b015f8afce44e4e02d8b908ed857afbd1

haval192,4 48 ae73833a09e84691d0214f360ee5027396f12599e3618118

haval224,4 56 e1ad67dc7a5901496b15dab92c2715de4b120af2baf661ecd92

haval256,4 64 2d39577df3a6a63168826b2a10f07a65a676f5776a0772e0a87

haval128,5 32 d20e920d5be9d9d34855accb501d1987

haval160,5 40 dac5e2024bfea142e53d1422b90c9ee2c8187cc6

haval192,5 48 bbb99b1e989ec3174019b20792fd92dd67175c2ff6ce5965

haval224,5 56 aa6551d75e33a9c5cd4141e9a068b1fc7b6d847f85c3ab16295

haval256,5 64 348298791817d5088a6de6c1b6364756d404a50bd64e645035f

**5. Fungsi**sha1()

Fungsi ini akan menghasilkan kode *hash* sepanjang 40 karakter. Mirip seperti fungsi md5().

Contoh:

<?php echo sha1("petanikode"); ?>

Hasilnya:

2e71f75dee7cb3e4b8e2e6ef8b8b5e0eab8f78a2

Seperti halnya fungsi md5(), fungsi ini juga akan menciptakan karakter aneh, bila ditambahkan garam.

Contoh:

<?php echo sha1("petanikode", "ga12am"); ?>

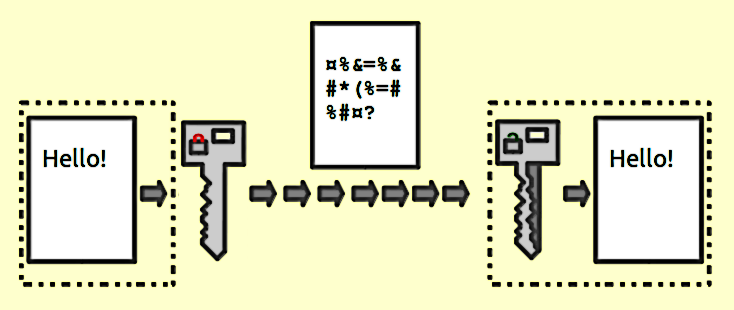
Hasilnya:

.q�]�|�����^��

**6. Fungsi**base64\_encode()

Fungsi ini akan menghasilkan kode *hash* dari teks yang diinputkan dan bisa dikembalikan ke bentuk semula dengan fungsi base64\_decode().

Metode ini disebut *two-way hasing*.



Contoh:

<?php echo base64\_encode("Petani kode"); ?>

Hasilnya:

UGV0YW5pIGtvZGU=

Sementara itu, untuk mengembalikan *(decrypt)* atau *decode* dapat menggunakan fungsi base64\_decode().

Contoh:

<?php echo base64\_decode("UGV0YW5pIGtvZGU="); ?>

Hasilnya:

Petani kode

Enkripsi dengan base64 tidak cocok digunakan untuk mengenkripsi password, karena sangat mudah di-decode.

**Akhir Kata…**

Itulah 6 (enam) fungsi enkripsi di PHP untuk mengamankan data. Setiap fungsi memiliki algoritmanya masing-masing.

Gunakan fungsi yang menggunakan metode *one-way hasing* untuk melindungi password. Tapi, perlu diselidiki dulu…apakah fungsi tersebut masih aman atau tidak?

Sedangkan untuk fungsi yang menggunakan metode *two-way hasing*, sangat tidak disarankan untuk melindungi password, karena memiliki fungsi untuk *decode* atau mengembalikannya dari bentuk *hash* menjadi *plain text*.