Funció crypt()

La funció `crypt()` de PHP serveix per generar un **hash unidireccional (irreversible)** d'una cadena de caràcters. Històricament, és la funció original d'estil Unix per crear hashes de contrasenyes. Actualment, segueix existint a PHP, **es recomana utilitzar** `password_hash()` en el seu lloc per garantir la màxima seguretat.

Què fa exactament `crypt()`

La funció `crypt()` pren una cadena d'entrada (la contrasenya) i una **sal** (salt) opcional (o obligatòria en versions modernes de PHP) i produeix un hash basat en l'algoritme que la sal especifiqui.

La sintaxi bàsica és:

PHP

```
crypt(string $string, string $salt): string
```

Com funciona Salt

La característica distintiva de `crypt()` és que la **sal no només afegeix aleatorietat al hash, sinó que també defineix quin algoritme de hashing utilitzarà**.

Depenent de com comenci la cadena de sal, `crypt()` pot utilitzar diferents algorismes:

Inici de la Sal	Algoritme	Característica
Sense prefaci (2 caràcters)	DES Estàndard	Molt antic i **insegur**.
`$1$`	MD5	Més antic i poc segur per a contrasenyes.
`$2y$` **o** `$2a$`	Blowfish (Bcrypt)	Fort i adequat per a contrasenyes.
`$5$`	SHA-256	Algoritme fort, però sense la funció de "cost" ajustable de Bcrypt.
`$6$`	SHA-512	Similar a SHA-256.

Cal utilitzar `password_hash()`

Tot i que `crypt()` pot utilitzar algorismes segurs com Blowfish, té greus inconvenients de gestió que fan que `password_hash()` sigui la millor opció:

Característica	`crypt()`	`password_hash()`
Gestió de la Sal	L'usuari ha de **generar manualment la sal** i incloure-hi el prefaci de l'algoritme (`$2y$`, `$5$`, etc.).	Genera automàticament una **sal forta i única** cada vegada, i la insereix al hash.
Algoritme (Per Defecte)	Per defecte, pot caure en algorismes febles com DES si la sal no és correcta.	Per defecte, utilitza **Bcrypt** (`PASSWORD_DEFAULT`), que és lent i segur.
Resistència al Futur	No té mecanisme d'actualització.	Permet utilitzar `password_needs_rehash()` per actualitzar automàticament el hash a algorismes més nous sense que l'usuari hagi de canviar la contrasenya.
Compatibilitat	El hash resultant és compatible amb `password_verify()`.	Simplifica tota l'operació en una sola funció moderna.

> `password_hash()` és simplement un **"embolcall" segur i modern** de `crypt()` que automatitza i simplifica les bones pràctiques de seguretat, evitant que els desenvolupadors cometin errors en la gestió de la sal i l'algoritme.

Exemple

```
<?php
$contrasenya = "Password123";

// 1. Ús de l'Algoritme ANTIC: DES Estàndard de Unix
// La sal és una cadena de 2 caràcters que no comença amb un prefix ($).
$salt_des = "ab";
$hash_des = crypt($contrasenya, $salt_des);

echo "--- Algoritme DES (Antic) ---\n";
echo "Sal utilitzada: " . $salt_des . "\n";
echo "Hash (8 caràcters de contrasenya utilitzats): " . $hash_des . "\n\n";

// 2. Ús de l'Algoritme MODERN: Blowfish (Bcrypt)
// El prefix '$2y$' força l'ús de Bcrypt i la resta de la cadena (22 caràcters) és la
sal.
$salt_bcrypt = '$2y$10$UnSaltDe22CaractersAci';
$hash_bcrypt = crypt($contrasenya, $salt_bcrypt);

echo "--- Algoritme Blowfish (Bcrypt) ---\n";
echo "Sal (i prefix) utilitzada: " . $salt_bcrypt . "\n";
echo "Hash resultant (60 caràcters): " . $hash_bcrypt . "\n\n";

// 3. Verificació (Utilitzant el hash generat)
$intent_correcte = "Password123";
$intent_incorrecte = "Malson123";
```

```
// Per verificar, s'utilitza la funció password_verify() per seguretat (fins i tot amb hashes de crypt()).
if (password_verify($intent_correcte, $hash_bcrypt)) {
    echo "Verificació Correcta: La contrasenya coincideix. \n";
} else {
    echo "Verificació Fallida: Error de comprovació. \n";
}
?>
```

Revisió núm. 1
★Admin l'ha creat 2025-10-14 10:08:08 UTC
Admin l'ha actualitzat 2025-10-14 10:14:01 UTC