ARYATECHNO

# Laravel - Hashing

**Topics :** Laravel
**Written on** December 25, 2023

In Laravel, hashing is the process of securely transforming sensitive data, such as passwords, into a fixed-length string of characters. Laravel uses the Bcrypt hashing algorithm by default, which is a secure one-way hashing algorithm.

## 1. Hashing a Value:

To hash a value, such as a password, you can use the `bcrypt` helper function or the `Hash` facade.

**Using `bcrypt` helper function:**

```
$hashedValue = bcrypt('secret_password');
```

**Using `Hash` facade:**

```
use Illuminate\Support\Facades\Hash;
```

$hashedValue = Hash::make('secret_password');

## 2. Verifying a Hashed Value:

To verify a hashed value, you can use the `Hash::check` method.

```
use Illuminate\Support\Facades\Hash;

$hashedPassword = Hash::make('secret_password');
$isMatch = Hash::check('secret_password', $hashedPassword);

if ($isMatch) {
    // Password is correct
} else {
    // Password is incorrect
}
```

## 3. Using Hashing in Eloquent Models:

In Eloquent models, you can leverage the `Hash` facade to automatically hash attributes when setting them.

```
use Illuminate\Support\Facades\Hash;

class User extends Model
{
    protected $fillable = ['name', 'email', 'password'];

    public function setPasswordAttribute($value)
    {
        $this->attributes['password'] = Hash::make($value);
    }
}
```

## 4. Customizing Hashing Configurations:

You can customize hashing configurations, such as the hashing algorithm and cost factor, in the `config/hashing.php` configuration file.

```
return [
    'driver' => 'bcrypt',
    'bcrypt' => [
        'rounds' => 10,
    ],
];
```

## 5. Using a Different Hashing Algorithm:

While Bcrypt is the default hashing algorithm, Laravel supports other algorithms like Argon2. You can configure this in the `config/hashing.php` file.

## 6. Checking if a Value Needs Rehashing:

In case you update your hashing algorithm or configuration, you can use the `Hash::needsRehash` method to check if a hashed value needs rehashing.

```
use Illuminate\Support\Facades\Hash;

if (Hash::needsRehash($hashedValue)) {
    $newHashedValue = Hash::make('secret_password');
    // Save $newHashedValue to the database
}
```

## 7. Hashing User Passwords in Authentication:

Laravel's authentication system automatically handles hashing and verifying user passwords.

```
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Hash;

// Attempt to authenticate the user
if (Auth::attempt(['email' => $email, 'password' => $password])) {
    // The user is authenticated
}
```

### 8. Hashing Passwords in Forms:

When working with forms, you can hash passwords before sending them to the server.

```
<form method="POST" action="/login">
    @csrf
    <input type="text" name="email" required>
    <input type="password" name="password" required>
    <button type="submit">Login</button>
</form>
```

### 9. Hashing API Tokens:

Laravel Passport automatically hashes API tokens.

### 10. Clearing Hashed Values:

If you need to clear hashed values (e.g., when seeding the database), you can use the `Hash::make` method with a fixed seed value.

```
use Illuminate\Support\Facades\Hash;

$hashedValue = Hash::make('secret_password', ['rounds' => 4]);
```