Encryption

Introduction

Laravel's encrypter uses OpenSSL to provide AES-256 and AES-128 encryption. You are strongly encouraged to use Laravel's built-in encryption facilities and not attempt to roll your own "home grown" encryption algorithms. All of Laravel's encrypted values are signed using a message authentication code (MAC) so that their underlying value can not be modified once encrypted.

Configuration

Before using Laravel's encrypter, you must set a `key` option in your `config/app.php` configuration file. You should use the `php artisan key:generate` command to generate this key since this Artisan command will use PHP's secure random bytes generator to build your key. If this value is not properly set, all values encrypted by Laravel will be insecure.

Using The Encrypter

Encrypting A Value

You may encrypt a value using the `encrypt` helper. All encrypted values are encrypted using OpenSSL and the `AES-256-CBC` cipher. Furthermore, all encrypted values are signed with a message authentication code (MAC) to detect any modifications to the encrypted string:

```php
<?php

namespace App\Http\Controllers;

use App\Http\Controllers\Controller;
use App\User;
use Illuminate\Http\Request;

class UserController extends Controller
{
    /**
     * Store a secret message for the user.
     *
     * @param  Request  $request
     * @param  int  $id
     * @return Response
     */
    public function storeSecret(Request $request, $id)
    {
        $user = User::findOrFail($id);

        $user->fill([
            'secret' => encrypt($request->secret),
        ])->save();
    }
}
```

Encrypting Without Serialization

Encrypted values are passed through `serialize` during encryption, which allows for encryption of objects and arrays. Thus, non-PHP clients receiving encrypted values will need to `unserialize` the data. If you would like to encrypt and decrypt values without serialization, you may use the `encryptString` and `decryptString` methods of the `Crypt` facade:

```php
use Illuminate\Support\Facades\Crypt;

$encrypted = Crypt::encryptString('Hello world.');

$decrypted = Crypt::decryptString($encrypted);
```

Decrypting A Value

You may decrypt values using the `decrypt` helper. If the value can not be properly decrypted, such as when the MAC is invalid, an `Illuminate\Contracts\Encryption\DecryptException` will be thrown:

```php
use Illuminate\Contracts\Encryption\DecryptException;

try {
    $decrypted = decrypt($encryptedValue);
} catch (DecryptException $e) {
    //
}
```

Become a Laravel Partner

Laravel Partners are elite shops providing top-notch Laravel development and consulting. Each of our partners can help you craft a beautiful, well-architected project.

Our Partners

Laravel

Highlights

- Release Notes
- Getting Started
- Routing
- Blade Templates
- Authentication
- Authorization
- Artisan Console
- Database
- Eloquent ORM
- Testing

Resources

- Laracasts
- Laravel News
- Laracon
- Laracon EU
- Laracon AU
- Jobs
- Certification
- Forums

Partners

- Vehikl
- Tighten Co.
- Kirschbaum
- Byte 5
- 64Robots
- Cubet
- DevSquad
- Ideil
- Cyber-Duck
- ABOUT YOU
- Become A Partner

Ecosystem

- Vapor
- Forge
- Envoyer
- Horizon
- Lumen
- Nova
- Echo
- Valet
- Mix
- Spark
- Cashier
- Homestead
- Dusk
- Passport
- Scout
- Socialite

Laravel is a web application framework with expressive, elegant syntax. We believe development must be an enjoyable and creative experience to be truly fulfilling. Laravel attempts to take the pain out of development by easing common tasks used in most web projects.