

Introduction

PHP library to load, validate and generate digital certificates, with keys, self-signed certificates and a service

Anonymous · 10/10/2026

Library for digital certificates

github derafu/certificate last commit august  CI passing code size 72.7 KiB open issues 0

downloads 6.55 k downloads 1.17 k this month

A comprehensive PHP library for working with digital certificates, providing tools for loading, validating and generating certificates.

Features

- **Certificate Loading:** Load certificates from files, data, arrays or keys.
- **Certificate Validation:** Validate certificates against specific requirements.
- **Certificate Generation:** Create self-signed certificates for testing.
- **Certificate Information:** Extract key information from certificates (ID, name, email, etc.).
- **Key Management:** Work with public and private keys, modulus, and exponent.

Installation

```
composer require derafu/certificate
```

Basic Usage

Loading a Certificate

```
use Derafu\Certificate\Service\CertificateLoader;

// Create a loader.
$loader = new CertificateLoader();

// Load from a file.
$certificate = $loader->loadFromFile('/path/to/certificate.p12', 'password');

// Load from data.
```

```

$certificate = $loader->loadFromData($certificateData, 'password');

// Load from array.
$certificate = $loader->loadFromArray([
    'cert' => $publicKey,
    'pkey' => $privateKey
]);

// Load from keys.
$certificate = $loader->loadFromKeys($publicKey, $privateKey);

```

Accessing Certificate Information

```

// Get basic certificate information.
$id = $certificate->getId(); // e.g., "12345678-9"
$name = $certificate->getName(); // e.g., "John Doe"
$email = $certificate->getEmail(); // e.g., "john.doe@example.com"

// Check certificate validity.
$isActive = $certificate->isActive(); // true if certificate is valid.
$expirationDays = $certificate->getExpirationDays(); // days until expiration.

// Get validity dates.
$validFrom = $certificate->getFrom(); // e.g., "2025-01-01T00:00:00"
$validTo = $certificate->getTo(); // e.g., "2026-01-01T00:00:00"

// Get certificate issuer.
$issuer = $certificate->getIssuer(); // e.g., "Example CA"

// Get key components.
$modulus = $certificate->getModulus();
$exponent = $certificate->getExponent();

// Get raw keys.
$publicKey = $certificate->getPublicKey(); // with headers.
$privateKey = $certificate->getPrivateKey(); // with headers.
$cleanPublicKey = $certificate->getPublicKey(true); // without headers.
$cleanPrivateKey = $certificate->getPrivateKey(true); // without headers.

```

Validating a Certificate

```

use Derafu\Certificate\Exception\CertificateException;
use Derafu\Certificate\Service\CertificateValidator;

$validator = new CertificateValidator();

try {
    $validator->validate($certificate);
    echo "Certificate is valid";
}

```

```
} catch (CertificateException $e) {  
    echo "Certificate validation failed: " . $e->getMessage();  
}
```

Creating a Fake Certificate for Testing

```
use Derafu\Certificate\Service\CertificateLoader;  
use Derafu\Certificate\Service\CertificateFaker;  
  
$loader = new CertificateLoader();  
$faker = new CertificateFaker($loader);  
  
// Create a fake certificate with default values.  
$certificate = $faker->createFake();  
  
// Create a fake certificate with custom values.  
$certificate = $faker->createFake(  
    id: '12345678-9',  
    name: 'John Doe',  
    email: 'john.doe@example.com',  
    password: 'secure_password'  
);  
  
// Export to PKCS#12 format.  
$pkcs12Data = $certificate->getPkcs12('password');  
file_put_contents('certificate.p12', $pkcs12Data);
```

Using the Service

The `CertificateService` provides a unified interface to all library functionality:

```
use Derafu\Certificate\Service\CertificateLoader;  
use Derafu\Certificate\Service\CertificateFaker;  
use Derafu\Certificate\Service\CertificateValidator;  
use Derafu\Certificate\Service\CertificateService;  
  
// Create the service with its dependencies.  
$loader = new CertificateLoader();  
$faker = new CertificateFaker($loader);  
$validator = new CertificateValidator();  
$service = new CertificateService($faker, $loader, $validator);  
  
// Use the service for certificate operations.  
$certificate = $service->loadFromFile('/path/to/certificate.p12', 'password');  
$service->validate($certificate);  
  
// Create a fake certificate for testing.  
$fakeCertificate = $service->createFake()
```

```
'12345678-9',  
'John Doe',  
'john.doe@example.com'  
);
```

Advanced Usage

Creating a Self-Signed Certificate

For more control over certificate generation:

```
use Derafu\Certificate\SelfSignedCertificate;  
use Derafu\Certificate\Service\CertificateLoader;  
  
// Create a self-signed certificate with custom values.  
$selfSigned = new SelfSignedCertificate();  
$selfSigned->setSubject(  
    C: 'US',  
    ST: 'California',  
    L: 'San Francisco',  
    O: 'Example Organization',  
    OU: 'IT Department',  
    CN: 'John Doe',  
    emailAddress: 'john.doe@example.com',  
    serialNumber: '12345678-9'  
);  
  
$selfSigned->setIssuer(  
    CN: 'Example CA'  
);  
  
$selfSigned->setValidity(365); // Valid for 1 year.  
$selfSigned->setPassword('secure_password');  
  
// Get the certificate array.  
$certArray = $selfSigned->toArray();  
  
// Load as a Certificate object.  
$loader = new CertificateLoader();  
$certificate = $loader->loadFromArray($certArray);
```

Working with Asymmetric Keys

```
use Derafu\Certificate\AsymmetricKeyHelper;  
  
// Normalize a public key (add headers if missing).  
$normalizedPublicKey = AsymmetricKeyHelper::normalizePublicKey($rawPublicKey);
```

```
// Normalize a private key (add headers if missing).
$normalizedPrivateKey = AsymmetricKeyHelper::normalizePrivateKey($rawPrivateKey);

// Generate a public key from modulus and exponent.
// Requirements: composer require phpseclib/phpseclib
$publicKey = AsymmetricKeyHelper::generatePublicKeyFromModulusExponent(
    $modulus,
    $exponent
);
```

Last updated on 10/10/2026