Signature Generator

SignatureGenerator Class

Anonymous · 09/09/2026

SignatureGenerator Class

The `SignatureGenerator` class is responsible for creating digital signatures, both for general data and specifically for XML documents according to the XML-DSIG standard.

Overview

The `SignatureGenerator` implements the `SignatureGeneratorInterface` and provides mechanisms to:

1. Sign any data using a private key.
2. Sign XML documents using a certificate.
3. Calculate digest values for XML documents or specific elements within them.

This class is a core component of the Derafu Signature library and is typically used through the `SignatureService` facade.

Basic Usage

Initialization

```
use Derafu\Signature\Service\SignatureGenerator;
use Derafu\Xml\Service\XmlDecoder;
use Derafu\Xml\Service\XmlEncoder;
use Derafu\Xml\Service\XmlService;
use Derafu\Xml\Service\XmlValidator;

// Initialize required XML services.
$xmlEncoder = new XmlEncoder();
$xmlDecoder = new XmlDecoder();
$xmlValidator = new XmlValidator();
$xmlService = new XmlService($xmlEncoder, $xmlDecoder, $xmlValidator);

// Create the signature generator.
$generator = new SignatureGenerator($xmlService);
```

Signing Data

```
// Sign data with a private key.
$data = 'Data to be signed';
$signature = $generator->sign($data, $privateKey);

// Optional: Specify a different signature algorithm.
$signature = $generator->sign($data, $privateKey, OPENSSL_ALGO_SHA256);
```

Signing XML Documents

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

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

// Sign an XML string.
$xmlString = '<root><element>data</element></root>';
$signedXml = $generator->signXml($xmlString, $certificate);

// Sign a specific element in the XML (identified by ID).
$xmlWithIds = '<root><element ID="myElement">data</element></root>';
$signedXml = $generator->signXml($xmlWithIds, $certificate, 'myElement');
```

Calculating Digest Values

```
// Create an XML document.
$xmlDoc = new \Derafu\Xml\XmlDocument();
$xmlDoc->loadXml('<root><element ID="myElement">data</element></root>');

// Calculate digest value for the entire document.
$digestValue = $generator->generateXmlDigestValue($xmlDoc);

// Calculate digest value for a specific element.
$digestValue = $generator->generateXmlDigestValue($xmlDoc, 'myElement');
```

API Reference

Data Signing

```
/**
 * Sign the provided data using a private key.
 *
 * @param string $data Data to be signed.
 * @param string $privateKey Private key to be used for signing.
```

```
     * @param string|int $signatureAlgorithm Algorithm to be used for signing (default
    SHA1).
     * @return string Digital signature in base64.
     * @throws SignatureException If the signing operation fails.
     */
    public function sign(
        string $data,
        string $privateKey,
        string|int $signatureAlgorithm = OPENSSL_ALGO_SHA1
    ): string;
```

XML Signing

```
    /**
     * Sign an XML document using RSA and SHA1.
     *
     * @param XmlDocumentInterface|string $xml XML document to be signed.
     * @param CertificateInterface $certificate Digital certificate to be used for
    signing.
     * @param ?string $reference Reference to which the signature is made. If not
     * specified, the digest of the entire XML document will be signed.
     * @return string String XML with the generated signature included in the
     * "Signature" tag at the end of the XML (last element within the root node).
     * @throws SignatureException If any problem occurs while signing.
     */
    public function signXml(
        XmlDocumentInterface|string $xml,
        CertificateInterface $certificate,
        ?string $reference = null
    ): string;
```

Digest Value Generation

```
    /**
     * Generate the SHA1 ("DigestValue") of a node of the XML with a certain
     * reference. This can be used later to generate the XML signature.
     *
     * If no reference is specified, the "DigestValue" will be calculated over the
     * entire XML (root node).
     *
     * @param XmlDocumentInterface $doc XML document to be signed.
     * @param ?string $reference Reference to which the signature is made.
     * @return string Data of the XML that must be digested.
     * @throws XmlException If the reference is not found in the XML.
     */
    public function generateXmlDigestValue(
        XmlDocumentInterface $doc,
        ?string $reference = null
    ): string;
```

Implementation Details

Digital Signature Process

For general data, the signing process is straightforward:

1. The data is signed using the private key and the specified algorithm.
2. The resulting binary signature is base64-encoded.
3. The base64-encoded signature is returned.

XML Signature Process

For XML documents, the signing process follows the XML-DSIG standard:

1. The XML document is loaded and parsed.
2. If a reference is provided, the referenced element is located.
3. The digest value (SHA1 hash) of the canonicalized (C14N) content is calculated.
4. A `Signature` node is created with the digest value and certificate information.
5. The `SignedInfo` element of the signature is canonicalized.
6. The canonicalized `SignedInfo` is signed using the certificate's private key.
7. The signature value is added to the `SignatureValue` element.
8. The complete signature node is added to the XML document.
9. The signed XML document is returned.

Canonicalization

The class uses the `C14NWithIso88591Encoding` method for canonicalization, which:

1. Applies XML canonicalization (C14N) according to the W3C standard.
2. Converts the result to ISO-8859-1 encoding.
3. Ensures consistent representation across different systems.

This process is crucial for ensuring that the same signature is generated regardless of the XML document's formatting or encoding.

Reference Handling

When a reference is provided:

1. The reference must be an ID attribute value in the XML document.
2. The digest is calculated only for the referenced element.
3. The signature's `Reference` element includes a URI attribute (`#elementId`).
4. The transform algorithm is set to standard C14N.

When no reference is provided:

1. The digest is calculated for the entire document (excluding any existing signature).
2. The signature's `Reference` element has an empty URI attribute.
3. The transform algorithm is set to "enveloped signature transformation".

Signature Node Creation

The class creates a `Signature` node with the following components:

1. `SignedInfo`: Contains information about what data was signed.
    - `CanonicalizationMethod`: Specifies the C14N algorithm.
    - `SignatureMethod`: Specifies RSA-SHA1.
    - `Reference`: Points to the signed content and includes the digest value.
2. `SignatureValue`: Contains the actual signature of the `SignedInfo` element.
3. `KeyInfo`: Contains information about the certificate used for signing.
    - `KeyValue/RSAKeyValue`: Contains the modulus and exponent from the certificate.
    - `X509Data/X509Certificate`: Contains the certificate itself.

Last updated on 09/09/2026