XML-DSIG Standard

XML Digital Signature (XML-DSIG)

Anonymous · 09/09/2026

XML Digital Signature (XML-DSIG)

This document provides an in-depth explanation of how the Derafu Signature library implements the XML Digital Signature (XML-DSIG) standard, including the structure of signature elements, the signing process, and the validation process.

XML-DSIG Overview

XML Digital Signatures provide integrity, message authentication, and signer authentication for XML data. The XML-DSIG standard is defined by the W3C in the XML Signature Syntax and Processing specification.

Signature Structure

The Derafu Signature library creates XML signatures with the following structure:

```
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <CanonicalizationMethod
Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
    <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
    <Reference URI="#elementId">
      <Transforms>
        <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
      </Transforms>
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <DigestValue>base64EncodedDigestValue</DigestValue>
    </Reference>
  </SignedInfo>
  <SignatureValue>base64EncodedSignatureValue</SignatureValue>
  <KeyInfo>
    <KeyValue>
      <RSAKeyValue>
        <Modulus>base64EncodedModulus</Modulus>
        <Exponent>base64EncodedExponent</Exponent>
      </RSAKeyValue>
    </KeyValue>
```

```
    <X509Data>
      <X509Certificate>base64EncodedCertificate</X509Certificate>
    </X509Data>
  </KeyInfo>
</Signature>
```

Key Components

1. **SignedInfo**: Contains information about what data is being signed.
    - **CanonicalizationMethod**: Specifies how the XML is normalized before signing.
    - **SignatureMethod**: Specifies the algorithm used for signing (RSA-SHA1).
    - **Reference**: Points to the data being signed.
        - **Transforms**: Describes transformations applied to the data before digesting.
        - **DigestMethod**: Specifies the algorithm used for the digest (SHA1).
        - **DigestValue**: Contains the base64-encoded digest of the data.
2. **SignatureValue**: Contains the base64-encoded signature of the canonicalized SignedInfo element.
3. **KeyInfo**: Contains information about the key used to validate the signature.
    - **KeyValue/RSAKeyValue**: Contains the RSA key parameters (modulus and exponent).
    - **X509Data/X509Certificate**: Contains the X.509 certificate used for signing.

Signing Process

The Derafu Signature library implements XML signing following these steps:

1. Preparing the Data

- If a reference ID is specified, the referenced element is located in the XML document.
- Otherwise, the entire XML document is used (excluding any existing Signature elements).

2. Calculating the Digest Value

- The data is canonicalized using the C14N algorithm.
- The canonicalized data is converted to ISO-8859-1 encoding.
- The SHA1 digest of the data is calculated and base64-encoded.

3. Creating the SignedInfo Element

- The SignedInfo element is created with the appropriate CanonicalizationMethod, SignatureMethod, and Reference elements.

- The DigestValue is included in the Reference element.

4. Calculating the Signature Value

- The SignedInfo element is canonicalized and converted to ISO-8859-1 encoding.
- The canonicalized SignedInfo is signed using the private key from the certificate.
- The resulting signature is base64-encoded and included in the SignatureValue element.

5. Including Key Information

- The public key components (modulus and exponent) are extracted from the certificate.
- The certificate itself is included in the X509Certificate element.

6. Adding the Signature to the Document

- The complete Signature element is added to the XML document, typically as the last child of the root element.

Validation Process

The Derafu Signature library validates XML signatures following these steps:

1. Locating Signature Elements

- All Signature elements in the XML document are located.
- Each signature is validated independently.

2. Validating the Digest Value

- The reference in the signature is extracted.
- The referenced data (or the entire document) is canonicalized and converted to ISO-8859-1 encoding.
- The SHA1 digest of the data is calculated and base64-encoded.
- The calculated digest is compared to the DigestValue in the signature.
- If they don’t match, the content integrity check fails.

3. Validating the Signature Value

- The SignedInfo element is canonicalized and converted to ISO-8859-1 encoding.
- The X.509 certificate is extracted from the X509Certificate element.
- The signature value is validated using the canonicalized SignedInfo, the SignatureValue, and the public key from the certificate.
- If the validation fails, the signer authenticity check fails.

Reference Types

The Derafu Signature library supports two types of references:

1. Element References

- Specified by a URI attribute with a value starting with `#` (e.g., `URI="#elementId"`).
- Only the referenced element is signed.
- The transform algorithm is set to standard C14N.

2. Whole Document References

- Specified by an empty URI attribute (`URI=""`).
- The entire document is signed, excluding any Signature elements.
- The transform algorithm is set to “enveloped signature transformation”.

Canonicalization

The Derafu Signature library uses the following canonicalization algorithms:

1. XML Canonicalization (C14N)

- Algorithm: `http://www.w3.org/TR/2001/REC-xml-c14n-20010315`
- Ensures consistent XML representation regardless of formatting differences.
- Applied to data before calculating digests and to SignedInfo before calculating signatures.

2. Enveloped Signature Transform

- Algorithm: `http://www.w3.org/2000/09/xmldsig#enveloped-signature`
- Excludes the signature itself when signing the entire document.
- Prevents circular references in the signing process.

Working with References and IDs

When using element references, the referenced element must have an `ID` attribute:

```
<root>
  <element ID="myElement">data</element>
</root>
```

The reference in the signature would be:

```
<Reference URI="#myElement">
```

```
  <!-- ... -->
</Reference>
```

The `SignatureGenerator.signXml()` method accepts the ID as a parameter:

```
$signedXml = $signatureGenerator->signXml($xml, $certificate, 'myElement');
```

Security Considerations

Digest Algorithm

The library uses SHA1 for digest calculation. While SHA1 is considered cryptographically weak for certain applications, it remains the standard algorithm specified in the XML-DSIG specification.

Signature Algorithm

The library uses RSA-SHA1 for signature calculation, which is the standard algorithm specified in the XML-DSIG specification.

Certificate Handling

The library includes the entire X.509 certificate in the signature, which allows for complete validation including certificate chain verification (although this is not currently implemented in the library).

Compatibility

The XML signatures generated by the Derafu Signature library follow the W3C XML-DSIG standard and should be compatible with other XML-DSIG implementations. However, different implementations may have subtle differences in canonicalization or other aspects of the signing process.

Common Issues

Invalid References

If a reference ID is specified but doesn't exist in the XML document, the signing process will fail.

Malformed XML

If the XML document is not well-formed, both signing and validation will fail.

Character Encoding

The library uses ISO-8859-1 encoding for canonicalized XML to ensure consistent digest and signature calculation across different systems.

Example Usage

Signing with a Reference

```
$xml = '<root><element ID="myElement">data</element></root>';
$signedXml = $signatureService->signXml($xml, $certificate, 'myElement');
```

Signing the Entire Document

```
$xml = '<root><element>data</element></root>';
$signedXml = $signatureService->signXml($xml, $certificate);
```

Validating a Signature

```
try {
    $signatureService->validateXml($signedXml);
    echo "Signature is valid!";
} catch (SignatureException $e) {
    echo "Signature validation failed: " . $e->getMessage();
}
```

Last updated on 09/09/2026