

Zend_Crypt_Rsa - Pádraic Brady

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

Zend Framework: Zend_Crypt_Rsa Component Proposal

Proposed Component Name	Zend_Crypt_Rsa
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Crypt_Rsa
Proposers	Pádraic Brady Matthew Weier O'Phinney, Zend Liaison
Revision	1.0 - 12 June 2008: Ready for review (wiki revision: 6)

Table of Contents

1. Overview
2. References
3. Component Requirements, Constraints, and Acceptance Criteria
4. Dependencies on Other Framework Components
5. Theory of Operation
6. Milestones / Tasks
7. Class Index
8. Use Cases
9. Class Skeletons

1. Overview

Zend_Crypt_Rsa is an implementation of the RSA algorithm for public-key cryptography. This proposal assumes a dependency on PHP's OpenSSL extension, though a later adapter would allow operation even in the absence of OpenSSL, or outside the limitations that OpenSSL imposes. The purpose of the proposal is to write an OO wrapper around the existing ext/openssl functionality where keys are represented as discrete objects rather than resources and strings. Both public and private keys utilise a common interface supporting common encryption, decryption, data signing and key generation and can be directly echoed as PEM strings for remote storage. The main input source at present is also PEM for both private and public keys, or X.509 certificates for public keys.

2. References

- [Ongoing implementation in Subversion with Unit Tests](#)

3. Component Requirements, Constraints, and Acceptance Criteria

- This component **will** utilise ext/openssl is available
- This component **will** support dual path encryption/decryption
- This component **will** support generating signed data signatures
- This component **will** generate key pairs for any bit-length > 384 bits

- This component **will** import PEM formatted keys and certificates
- This component "will not currently" import DER or PKCS#12 formats
- This component **will not currently** generate keys with less than 384 bits
- This component **will not currently** operate in the absence of ext/openssl

4. Dependencies on Other Framework Components

- Zend_Exception
- ext/openssl

5. Theory of Operation

The component operates in a simple manner. The Zend_Crypt_Rsa object is instantiated usually by passing the PEM representation string of a private key, from which both the private key itself, and the respective public key, are in turn instantiated as separate objects of type Zend_Crypt_Key. Alternatively passing in a PEM formatted X.509 certificate will instantiate the object only with a public key available. Key objects share a common interface, and can be echoed directly back into the PEM format.

Once instantiated the object is capable of performing a wide range of common RSA operations including signing data, key generation, encryption and decryption. Both encryption and decryption operations require users to explicitly pass the key to use for both processes, to facilitate either public key, or private key, based encryption, for instance where the original object was instantiated from a X.509 certificate and a private key will be passed in later.

6. Milestones / Tasks

- Milestone 1: [DONE] Assemble use cases and design comments based on draft source code
- Milestone 2: [DONE] Assemble a unit test suite
- Milestone 3: [UNDERWAY/PENDING COMMENT] Complete initial development
- Milestone 4: Verify unit test coverage
- Milestone 5: Write documentation

7. Class Index

- Zend_Crypt_Rsa
- Zend_Crypt_Rsa_Exception
- Zend_Crypt_Rsa_Key
- Zend_Crypt_Rsa_Key_Private
- Zend_Crypt_Rsa_Key_Public

This is a preliminary class listing. Pending public and Zend review, the list may expand as necessary.

8. Use Cases

UC-01

Simple key generation.

```
<?php

$rsrsa = new Zend_Crypt_Rsa;
$keys = $rsrsa->generateKeys(array('private_key_bits'=>512));
echo $keys->privateKey; // PEM string
echo $keys->publicKey; // PEM string
```

UC-02

Data signing for secure message exchanges.

```
<?php

$dataToSend = '1234567890';
$rsrsa = new Zend_Crypt_Rsa('/path/to/privatekey.pem');
$binarySignature = $rsrsa->sign($dataToSign);
$base64Signature = $rsrsa->sign($dataToSign, Zend_Crypt_Rsa::BASE64);
```

UC-03

Verifying received data using the signature also sent with message.

```
<?php

$dataReceived = '1234567890';
$signatureReceived = 'XXXXXXXXXXXXXXXXXXXXXXXXX'; //assume some binary signature
$rsrsa = new Zend_Crypt_Rsa('/path/to/privatekey.pem');
$result = $rsrsa->verifySignature($dataReceived, $signatureReceived);
var_dump($result); // TRUE or FALSE
```

UC-04

Public key encryption for public messaging of secure data.

```
<?php

$rsrsa = new Zend_Crypt_Rsa('/path/to/privatekey.pem');
$dataToEncrypt = '1234567890';
$encrypted = $rsrsa->encrypt($data, $rsrsa->getPublicKey());
```

UC-05

Decrypting UC-04 result.

```
<?php

$encrypted = 'XXXXXXXXXXXXXXXXXXXXXXXXX'; // assume encrypted data
$rsrsa = new Zend_Crypt_Rsa('/path/to/privatekey.pem');
$decrypted = $rsrsa->decrypt($data, $rsrsa->getPrivateKey());
```

UC-06

All keys can be queried for their PEM string forms, their respective OpenSSL Key resources, and bit length.

```
<?php

$rsrsa = new Zend_Crypt_Rsa('/path/to/privatekey.pem');
echo $rsrsa->getPublicKey(); // PEM string
echo count($rsrsa->getPrivateKey()); // bit count, class implemented Countable
echo get_resource_type($rsrsa->getPublicKey()->getOpensslKeyResource()); // Openssl key resource
```

UC-07

Public keys can be retrieved from X.509 certs

```
<?php
$rsrsa = new Zend_Crypt_Rsa(array('certificatePath'=>'path/to/x509.crt'));
$dataToEncrypt = '1234567890';
$encrypted = $rsrsa->encrypt($data, $rsrsa->getPublicKey());
$rsrsa2 = new Zend_Crypt_Rsa('/path/to/privkey.pem');
$decrypted = $rsrsa->decrypt($encrypted, $rsrsa2->getPrivateKey());
```

UC-08

Keys pairs can be generated to, or accessed from, PEM formats which are encrypted with a passphrase.

```
<?php
$rsrsa = new Zend_Crypt_Rsa(array('certificatePath'=>'path/to/x509.crt'));
$dataToEncrypt = '1234567890';
$encrypted = $rsrsa->encrypt($data, $rsrsa->getPublicKey());
$rsrsa2 = new Zend_Crypt_Rsa('/path/to/privkey.pem');
$decrypted = $rsrsa->decrypt($encrypted, $rsrsa2->getPrivateKey());
```

9. Class Skeletons

Source code and Unit Tests for this component are available from subversion (supports online viewing):
<http://svn.astrumfutura.com/zendframework/trunk>

```
]]></ac:plain-text-body></ac:macro>
]]></ac:plain-text-body></ac:macro>
```