PKCS#1 OAEP (RSA)

PKCS#1 OAEP is an asymmetric cipher based on RSA and the OAEP padding. It is described in RFC8017 where it is called `RSAES-OAEP`.

It can only encrypt messages slightly shorter than the RSA modulus (a few hundred bytes).

The following example shows how you encrypt data by means of the recipient's `public key` (here assumed to be available locally in a file called `public.pem`):

```
>>> from Crypto.Cipher import PKCS1_OAEP
>>> from Crypto.PublicKey import RSA
>>>
>>> message = b'You can attack now!'
>>> key = RSA.importKey(open('public.pem').read())
>>> cipher = PKCS1_OAEP.new(key)
>>> ciphertext = cipher.encrypt(message)
```

The recipient uses its own `private key` to decrypt the message. We assume the key is stored in a file called `private.pem`:

```
>>> key = RSA.importKey(open('private.pem').read())
>>> cipher = PKCS1_OAEP.new(key)
>>> message = cipher.decrypt(ciphertext)
```

Warning

PKCS#1 OAEP does not guarantee authenticity of the message you decrypt. Since the public key is not secret, everybody could have created the encrypted message. Asymmetric encryption is typically paired with a digital signature.

Note

This module does not generate nor load RSA keys. Refer to the `Crypto.PublicKey.RSA` module.

class **Crypto.Cipher.PKCS1_OAEP.PKCS1OAEP_Cipher**(*key*, *hashAlgo*, *mgfunc*, *label*, *randfunc*)

Cipher object for PKCS#1 v1.5 OAEP. Do not create directly: use `new()` instead.

can_decrypt()

Legacy function to check if you can call `decrypt()`.

Deprecated since version 3.0.

can_encrypt()

Legacy function to check if you can call `encrypt()`.

Deprecated since version 3.0.

decrypt(*ciphertext*)

Decrypt a message with PKCS#1 OAEP.

Parameters: ciphertext (*bytes/bytearray/memoryview*) – The encrypted message.

Returns: The original message (plaintext).

Return type: bytes

Raises:
- ValueError – if the ciphertext has the wrong length, or if decryption fails the integrity check (in which case, the decryption key is probably wrong).
- TypeError – if the RSA key has no private half (i.e. you are trying to decrypt using a public key).

encrypt(*message*)

Encrypt a message with PKCS#1 OAEP.

Parameters: message (*bytes/bytearray/memoryview*) – The message to encrypt, also known as plaintext. It can be of variable length, but not longer than the RSA modulus (in bytes) minus 2, minus twice the hash output size. For instance, if you use RSA 2048 and SHA-256, the longest message you can encrypt is 190 byte long.

Returns: The ciphertext, as large as the RSA modulus.

Return type: bytes

Raises: ValueError – if the message is too long.

Crypto.Cipher.PKCS1_OAEP.new(*key*, *hashAlgo=None*, *mgfunc=None*, *label=''*, *randfunc=None*)

Return a cipher object `PKCS1OAEP_Cipher` that can be used to perform PKCS#1 OAEP encryption or decryption.

Parameters:
- key (*RSA key object*) – The key object to use to encrypt or decrypt the message. Decryption is only possible with a private RSA key.
- hashAlgo (*hash object*) – The hash function to use. This can be a module under *Crypto.Hash* or an existing hash object created from any of such modules. If not

specified, *Crypto.Hash.SHA1* is used.

- `mgfunc` (*callable*) – A mask generation function that accepts two parameters: a string to use as seed, and the lenth of the mask to generate, in bytes. If not specified, the standard MGF1 consistent with `hashAlgo` is used (a safe choice).
- `label` (*bytes/bytearray/memoryview*) – A label to apply to this particular encryption. If not specified, an empty string is used. Specifying a label does not improve security.
- `randfunc` (*callable*) – A function that returns random bytes. The default is *Random.get_random_bytes*.