

NAME

RSA_padding_add_PKCS1_type_1, RSA_padding_check_PKCS1_type_1,
 RSA_padding_add_PKCS1_type_2, RSA_padding_check_PKCS1_type_2,
 RSA_padding_add_PKCS1_OAEP, RSA_padding_check_PKCS1_OAEP,
 RSA_padding_add_PKCS1_OAEP_mgf1, RSA_padding_check_PKCS1_OAEP_mgf1,
 RSA_padding_add_none, RSA_padding_check_none - asymmetric encryption padding

SYNOPSIS

```
#include <openssl/rsa.h>
```

The following functions have been deprecated since OpenSSL 3.0, and can be hidden entirely by defining **OPENSSL_API_COMPAT** with a suitable version value, see **openssl_user_macros(7)**:

```
int RSA_padding_add_PKCS1_type_1(unsigned char *to, int tlen,  
                                const unsigned char *f, int fl);
```

```
int RSA_padding_check_PKCS1_type_1(unsigned char *to, int tlen,  
                                   const unsigned char *f, int fl, int rsa_len);
```

```
int RSA_padding_add_PKCS1_type_2(unsigned char *to, int tlen,  
                                const unsigned char *f, int fl);
```

```
int RSA_padding_check_PKCS1_type_2(unsigned char *to, int tlen,  
                                   const unsigned char *f, int fl, int rsa_len);
```

```
int RSA_padding_add_PKCS1_OAEP(unsigned char *to, int tlen,  
                                const unsigned char *f, int fl,  
                                const unsigned char *p, int pl);
```

```
int RSA_padding_check_PKCS1_OAEP(unsigned char *to, int tlen,  
                                 const unsigned char *f, int fl, int rsa_len,  
                                 const unsigned char *p, int pl);
```

```
int RSA_padding_add_PKCS1_OAEP_mgf1(unsigned char *to, int tlen,  
                                     const unsigned char *f, int fl,  
                                     const unsigned char *p, int pl,  
                                     const EVP_MD *md, const EVP_MD *mgf1md);
```

```
int RSA_padding_check_PKCS1_OAEP_mgf1(unsigned char *to, int tlen,  
                                       const unsigned char *f, int fl, int rsa_len,
```

```

const unsigned char *p, int pl,
const EVP_MD *md, const EVP_MD *mgf1md);

int RSA_padding_add_none(unsigned char *to, int tlen,
const unsigned char *f, int fl);

int RSA_padding_check_none(unsigned char *to, int tlen,
const unsigned char *f, int fl, int rsa_len);

```

DESCRIPTION

All of the functions described on this page are deprecated. Applications should instead use the EVP PKEY APIs.

The **RSA_padding_XXX_XXX()** functions are called from the RSA encrypt, decrypt, sign and verify functions. Normally they should not be called from application programs.

However, they can also be called directly to implement padding for other asymmetric ciphers.

RSA_padding_add_PKCS1_OAEP() and **RSA_padding_check_PKCS1_OAEP()** may be used in an application combined with **RSA_NO_PADDING** in order to implement OAEP with an encoding parameter.

RSA_padding_add_XXX() encodes **fl** bytes from **f** so as to fit into **tlen** bytes and stores the result at **to**. An error occurs if **fl** does not meet the size requirements of the encoding method.

The following encoding methods are implemented:

PKCS1_type_1

PKCS #1 v2.0 EMSA-PKCS1-v1_5 (PKCS #1 v1.5 block type 1); used for signatures

PKCS1_type_2

PKCS #1 v2.0 EME-PKCS1-v1_5 (PKCS #1 v1.5 block type 2)

PKCS1_OAEP

PKCS #1 v2.0 EME-OAEP

none

simply copy the data

The random number generator must be seeded prior to calling **RSA_padding_add_XXX()**. If the automatic seeding or reseeding of the OpenSSL CSPRNG fails due to external circumstances (see

RAND(7)), the operation will fail.

RSA_padding_check_xxx() verifies that the **fl** bytes at **f** contain a valid encoding for a **rsa_len** byte RSA key in the respective encoding method and stores the recovered data of at most **tlen** bytes (for **RSA_NO_PADDING**: of size **tlen**) at **to**.

For **RSA_padding_xxx_OAEP()**, **p** points to the encoding parameter of length **pl**. **p** may be **NULL** if **pl** is 0.

For **RSA_padding_xxx_OAEP_mgf1()**, **md** points to the md hash, if **md** is **NULL** that means md=sha1, and **mgf1md** points to the mgf1 hash, if **mgf1md** is **NULL** that means mgf1md=md.

RETURN VALUES

The **RSA_padding_add_xxx()** functions return 1 on success, 0 on error. The

RSA_padding_check_xxx() functions return the length of the recovered data, -1 on error. Error codes can be obtained by calling **ERR_get_error(3)**.

WARNINGS

The result of **RSA_padding_check_PKCS1_type_2()** is a very sensitive information which can potentially be used to mount a Bleichenbacher padding oracle attack. This is an inherent weakness in the PKCS #1 v1.5 padding design. Prefer PKCS1_OAEP padding. If that is not possible, the result of **RSA_padding_check_PKCS1_type_2()** should be checked in constant time if it matches the expected length of the plaintext and additionally some application specific consistency checks on the plaintext need to be performed in constant time. If the plaintext is rejected it must be kept secret which of the checks caused the application to reject the message. Do not remove the zero-padding from the decrypted raw RSA data which was computed by **RSA_private_decrypt()** with **RSA_NO_PADDING**, as this would create a small timing side channel which could be used to mount a Bleichenbacher attack against any padding mode including PKCS1_OAEP.

SEE ALSO

RSA_public_encrypt(3), **RSA_private_decrypt(3)**, **RSA_sign(3)**, **RSA_verify(3)**, **RAND(7)**

HISTORY

All of these functions were deprecated in OpenSSL 3.0.

COPYRIGHT

Copyright 2000-2021 The OpenSSL Project Authors. All Rights Reserved.

Licensed under the Apache License 2.0 (the "License"). You may not use this file except in compliance with the License. You can obtain a copy in the file LICENSE in the source distribution or

at <<https://www.openssl.org/source/license.html>>.