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_SSLv23, RSA_padding_check_SSLv23, RSA_padding_add_none, RSA_padding_check_none - asymmetric encryption padding

SYNOPSIS

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

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

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

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

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

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

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

int RSA_padding_add_SSLv23(unsigned char *to, int tlen,
unsigned char *f, int fl);

int RSA_padding_check_SSLv23(unsigned char *to, int tlen,
unsigned char *f, int fl, int rsa_len);

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

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

DESCRIPTION

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

SSLv23
: PKCS #1 EME-PKCS1-v1_5 with SSL-specific modification

none
: simply copy the data

The random number generator must be seeded prior to calling *RSA_padding_add_xxx()*.

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.

RETURN VALUES

The *RSA_padding_add_xxx()* functions return 1 on success, 0 on error. The*RSA_padding_c heck_xxx()* functions return the length of the recovered data, -1 on error. Error codes can be obtained by calling *ERR_get_error(3)* .

WARNING

The *RSA_padding_check_PKCS1_type_2()* padding check leaks timing 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. Otherwise it can be recommended to pass zero-padded **f**, so that **fl** equals to **rsa_len**, and if fixed by protocol, **tlen** being set to the expected length. In such case leakage would be minimal, it would take attacker's ability to observe memory access pattern with byte granularity as it occurs, post-factum timing analysis won't do.

SEE ALSO

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

COPYRIGHT