

NAME

RSA_set_default_method, RSA_get_default_method, RSA_set_method, RSA_get_method,
 RSA_PKCS1_OpenSSL, RSA_flags, RSA_new_method – select RSA method

SYNOPSIS

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

void RSA_set_default_method(const RSA_METHOD *meth);

RSA_METHOD *RSA_get_default_method(void);

int RSA_set_method(RSA *rsa, const RSA_METHOD *meth);

RSA_METHOD *RSA_get_method(const RSA *rsa);

RSA_METHOD *RSA_PKCS1_OpenSSL(void);

int RSA_flags(const RSA *rsa);

RSA *RSA_new_method(ENGINE *engine);
```

DESCRIPTION

An **RSA_METHOD** specifies the functions that OpenSSL uses for RSA operations. By modifying the method, alternative implementations such as hardware accelerators may be used. **IMPORTANT:** See the NOTES section for important information about how these RSA API functions are affected by the use of **ENGINE** API calls.

Initially, the default **RSA_METHOD** is the OpenSSL internal implementation, as returned by **RSA_PKCS1_OpenSSL()**.

RSA_set_default_method() makes **meth** the default method for all RSA structures created later. **NB:** This is true only whilst no **ENGINE** has been set as a default for RSA, so this function is no longer recommended. This function is not thread-safe and should not be called at the same time as other OpenSSL functions.

RSA_get_default_method() returns a pointer to the current default **RSA_METHOD**. However, the meaningfulness of this result is dependent on whether the **ENGINE** API is being used, so this function is no longer recommended.

RSA_set_method() selects **meth** to perform all operations using the key **rsa**. This will replace the **RSA_METHOD** used by the RSA key and if the previous method was supplied by an **ENGINE**, the handle to that **ENGINE** will be released during the change. It is possible to have RSA keys that only work with certain **RSA_METHOD** implementations (e.g. from an **ENGINE** module that supports embedded hardware-protected keys), and in such cases attempting to change the **RSA_METHOD** for the key can have unexpected results.

RSA_get_method() returns a pointer to the **RSA_METHOD** being used by **rsa**. This method may or may not be supplied by an **ENGINE** implementation, but if it is, the return value can only be guaranteed to be valid as long as the RSA key itself is valid and does not have its implementation changed by **RSA_set_method()**.

RSA_flags() returns the **flags** that are set for **rsa**'s current **RSA_METHOD**. See the BUGS section.

RSA_new_method() allocates and initializes an RSA structure so that **engine** will be used for the RSA operations. If **engine** is **NULL**, the default **ENGINE** for RSA operations is used, and if no default **ENGINE** is set, the **RSA_METHOD** controlled by **RSA_set_default_method()** is used.

RSA_flags() returns the **flags** that are set for **rsa**'s current method.

RSA_new_method() allocates and initializes an **RSA** structure so that **method** will be used for the RSA operations. If **method** is **NULL**, the default method is used.

THE RSA_METHOD STRUCTURE

```

typedef struct rsa_meth_st
{
    /* name of the implementation */
    const char *name;

    /* encrypt */
    int (*rsa_pub_enc)(int flen, unsigned char *from,
                      unsigned char *to, RSA *rsa, int padding);

    /* verify arbitrary data */
    int (*rsa_pub_dec)(int flen, unsigned char *from,
                      unsigned char *to, RSA *rsa, int padding);

    /* sign arbitrary data */
    int (*rsa_priv_enc)(int flen, unsigned char *from,
                       unsigned char *to, RSA *rsa, int padding);

    /* decrypt */
    int (*rsa_priv_dec)(int flen, unsigned char *from,
                       unsigned char *to, RSA *rsa, int padding);

    /* compute  $r_0 = r_0 \wedge I \pmod{rsa \rightarrow n}$  (May be NULL for some implementations) */
    int (*rsa_mod_exp)(BIGNUM *r0, BIGNUM *I, RSA *rsa);

    /* compute  $r = a \wedge p \pmod{m}$  (May be NULL for some implementations) */
    int (*bn_mod_exp)(BIGNUM *r, BIGNUM *a, const BIGNUM *p,
                     const BIGNUM *m, BN_CTX *ctx, BN_MONT_CTX *m_ctx);

    /* called at RSA_new */
    int (*init)(RSA *rsa);

    /* called at RSA_free */
    int (*finish)(RSA *rsa);

    /*
     * RSA_FLAG_EXT_PKEY          - rsa_mod_exp is called for private key
     *                            operations, even if p,q,dmp1,dmql,iqmp
     *                            are NULL
     * RSA_METHOD_FLAG_NO_CHECK - don't check pub/private match
     */
    int flags;

    char *app_data; /* ?? */

    int (*rsa_sign)(int type,
                   const unsigned char *m, unsigned int m_length,
                   unsigned char *sigret, unsigned int *siglen, const RSA *rsa);
    int (*rsa_verify)(int dtype,
                     const unsigned char *m, unsigned int m_length,
                     const unsigned char *sigbuf, unsigned int siglen,
                     const RSA *rsa);

    /* keygen. If NULL builtin RSA key generation will be used */
    int (*rsa_keygen)(RSA *rsa, int bits, BIGNUM *e, BN_GENCB *cb);

```

```
} RSA_METHOD;
```

RETURN VALUES

RSA_PKCS1_OpenSSL(), **RSA_PKCS1_null_method()**, **RSA_get_default_method()** and **RSA_get_method()** return pointers to the respective RSA_METHODs.

RSA_set_default_method() returns no value.

RSA_set_method() returns a pointer to the old RSA_METHOD implementation that was replaced. However, this return value should probably be ignored because if it was supplied by an ENGINE, the pointer could be invalidated at any time if the ENGINE is unloaded (in fact it could be unloaded as a result of the **RSA_set_method()** function releasing its handle to the ENGINE). For this reason, the return type may be replaced with a **void** declaration in a future release.

RSA_new_method() returns NULL and sets an error code that can be obtained by **ERR_get_error(3)** if the allocation fails. Otherwise it returns a pointer to the newly allocated structure.

BUGS

The behaviour of **RSA_flags()** is a mis-feature that is left as-is for now to avoid creating compatibility problems. RSA functionality, such as the encryption functions, are controlled by the **flags** value in the RSA key itself, not by the **flags** value in the RSA_METHOD attached to the RSA key (which is what this function returns). If the flags element of an RSA key is changed, the changes will be honoured by RSA functionality but will not be reflected in the return value of the **RSA_flags()** function – in effect **RSA_flags()** behaves more like an **RSA_default_flags()** function (which does not currently exist).

SEE ALSO

[RSA_new\(3\)](#)

HISTORY

The **RSA_null_method()**, which was a partial attempt to avoid patent issues, was replaced to always return NULL in OpenSSL 1.1.1.

COPYRIGHT

Copyright 2000–2020 The OpenSSL Project Authors. All Rights Reserved.

Licensed under the OpenSSL license (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>.