
## *Rocky Enterprise Linux 9.2 Manual Pages on command 'RSA_set0_multi_prime_params.3ossl'*

***$ man RSA_set0_multi_prime_params.3ossl***

RSA_GET0_KEY(3ossl) OpenSSL RSA_GET0_KEY(3ossl)

NAME

RSA_set0_key, RSA_set0_factors, RSA_set0_crt_params, RSA_get0_key, RSA_get0_factors, RSA_get0_crt_params, RSA_get0_n, RSA_get0_e, RSA_get0_d, RSA_get0_p, RSA_get0_q, RSA_get0_dmp1, RSA_get0_dmq1, RSA_get0_iqmp, RSA_get0_pss_params, RSA_clear_flags, RSA_test_flags, RSA_set_flags, RSA_get0_engine, RSA_get_multi_prime_extra_count, RSA_get0_multi_prime_factors, RSA_get0_multi_prime_crt_params, RSA_set0_multi_prime_params, RSA_get_version - Routines for getting and setting data in an RSA object

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_set0_key(RSA *r, BIGNUM *n, BIGNUM *e, BIGNUM *d);

int RSA_set0_factors(RSA *r, BIGNUM *p, BIGNUM *q);

int RSA_set0_crt_params(RSA *r, BIGNUM *dmp1, BIGNUM *dmq1, BIGNUM *iqmp);

```
void RSA_get0_key(const RSA *r,
                  const BIGNUM **n, const BIGNUM **e, const BIGNUM **d);
void RSA_get0_factors(const RSA *r, const BIGNUM **p, const BIGNUM **q);
void RSA_get0_crt_params(const RSA *r,
                         const BIGNUM **dmp1, const BIGNUM **dmq1,
                         const BIGNUM **iqmp);
const BIGNUM *RSA_get0_n(const RSA *d);
const BIGNUM *RSA_get0_e(const RSA *d);
const BIGNUM *RSA_get0_d(const RSA *d);
const BIGNUM *RSA_get0_p(const RSA *d);
const BIGNUM *RSA_get0_q(const RSA *d);
const BIGNUM *RSA_get0_dmp1(const RSA *r);
const BIGNUM *RSA_get0_dmq1(const RSA *r);
const BIGNUM *RSA_get0_iqmp(const RSA *r);
const RSA_PSS_PARAMS *RSA_get0_pss_params(const RSA *r);
void RSA_clear_flags(RSA *r, int flags);
int RSA_test_flags(const RSA *r, int flags);
void RSA_set_flags(RSA *r, int flags);
ENGINE *RSA_get0_engine(RSA *r);
int RSA_get_multi_prime_extra_count(const RSA *r);
int RSA_get0_multi_prime_factors(const RSA *r, const BIGNUM *primes[]);
int RSA_get0_multi_prime_crt_params(const RSA *r, const BIGNUM *exps[],
                                    const BIGNUM *coeffs[]);
int RSA_set0_multi_prime_params(RSA *r, BIGNUM *primes[], BIGNUM *exps[],
                                BIGNUM *coeffs[], int pnum);
int RSA_get_version(RSA *r);
```

## DESCRIPTION

All of the functions described on this page are deprecated.

Applications should instead use EVP_PKEY_get_bn_param(3) for any methods that return a BIGNUM. Refer to EVP_PKEY-DH(7) for more infomation.

An RSA object contains the components for the public and private key, n, e, d, p, q, dmp1, dmq1 and iqmp. n is the modulus common to both

public and private key, e is the public exponent and d is the private exponent. p, q, dmp1, dmq1 and iqmp are the factors for the second representation of a private key (see PKCS#1 section 3 Key Types), where p and q are the first and second factor of n and dmp1, dmq1 and iqmp are the exponents and coefficient for CRT calculations.

For multi-prime RSA (defined in RFC 8017), there are also one or more 'triplet' in an RSA object. A triplet contains three members, r, d and t. r is the additional prime besides p and q. d and t are the exponent and coefficient for CRT calculations.

The n, e and d parameters can be obtained by calling RSA_get0_key(). If they have not been set yet, then *n, *e and *d will be set to NULL. Otherwise, they are set to pointers to their respective values. These point directly to the internal representations of the values and therefore should not be freed by the caller.

The n, e and d parameter values can be set by calling RSA_set0_key() and passing the new values for n, e and d as parameters to the function. The values n and e must be non-NULL the first time this function is called on a given RSA object. The value d may be NULL. On subsequent calls any of these values may be NULL which means the corresponding RSA field is left untouched. Calling this function transfers the memory management of the values to the RSA object, and therefore the values that have been passed in should not be freed by the caller after this function has been called.

In a similar fashion, the p and q parameters can be obtained and set with RSA_get0_factors() and RSA_set0_factors(), and the dmp1, dmq1 and iqmp parameters can be obtained and set with RSA_get0_crt_params() and RSA_set0_crt_params().

For RSA_get0_key(), RSA_get0_factors(), and RSA_get0_crt_params(), NULL value BIGNUM ** output parameters are permitted. The functions ignore NULL parameters but return values for other, non-NULL, parameters.

For multi-prime RSA, RSA_get0_multi_prime_factors() and RSA_get0_multi_prime_params() can be used to obtain other primes and related CRT parameters. The return values are stored in an array of

BIGNUM *. RSA_set0_multi_prime_params() sets a collect of multi-prime 'triplet' members (prime, exponent and coefficient) into an RSA object. Any of the values n, e, d, p, q, dmp1, dmq1, and iqmp can also be retrieved separately by the corresponding function RSA_get0_n(), RSA_get0_e(), RSA_get0_d(), RSA_get0_p(), RSA_get0_q(), RSA_get0_dmp1(), RSA_get0_dmq1(), and RSA_get0_iqmp(), respectively. RSA_get0_pss_params() is used to retrieve the RSA-PSS parameters.

RSA_set_flags() sets the flags in the flags parameter on the RSA object. Multiple flags can be passed in one go (bitwise ORed together). Any flags that are already set are left set. RSA_test_flags() tests to see whether the flags passed in the flags parameter are currently set in the RSA object. Multiple flags can be tested in one go. All flags that are currently set are returned, or zero if none of the flags are set. RSA_clear_flags() clears the specified flags within the RSA object.

RSA_get0_engine() returns a handle to the ENGINE that has been set for this RSA object, or NULL if no such ENGINE has been set.

RSA_get_version() returns the version of an RSA object r.

NOTES

Values retrieved with RSA_get0_key() are owned by the RSA object used in the call and may therefore not be passed to RSA_set0_key(). If needed, duplicate the received value using BN_dup() and pass the duplicate. The same applies to RSA_get0_factors() and RSA_set0_factors() as well as RSA_get0_crt_params() and RSA_set0_crt_params().

The caller should obtain the size by calling RSA_get_multi_prime_extra_count() in advance and allocate sufficient buffer to store the return values before calling RSA_get0_multi_prime_factors() and RSA_get0_multi_prime_params().

RSA_set0_multi_prime_params() always clears the original multi-prime triplets in RSA object r and assign the new set of triplets into it.

RETURN VALUES

RSA_set0_key(), RSA_set0_factors(), RSA_set0_crt_params() and

RSA_set0_multi_prime_params() return 1 on success or 0 on failure.

RSA_get0_n(), RSA_get0_e(), RSA_get0_d(), RSA_get0_p(), RSA_get0_q(), RSA_get0_dmp1(), RSA_get0_dmq1(), and RSA_get0_iqmp() return the respective value.

RSA_get0_pss_params() returns a RSA_PSS_PARAMS pointer, or NULL if there is none.

RSA_get0_multi_prime_factors() and RSA_get0_multi_prime_crt_params() return 1 on success or 0 on failure.

RSA_get_multi_prime_extra_count() returns two less than the number of primes in use, which is 0 for traditional RSA and the number of extra primes for multi-prime RSA.

RSA_get_version() returns RSA_ASN1_VERSION_MULTI for multi-prime RSA and RSA_ASN1_VERSION_DEFAULT for normal two-prime RSA, as defined in RFC 8017.

RSA_test_flags() returns the current state of the flags in the RSA object.

RSA_get0_engine() returns the ENGINE set for the RSA object or NULL if no ENGINE has been set.

## SEE ALSO

RSA_new(3), RSA_size(3)

## HISTORY

The RSA_get0_pss_params() function was added in OpenSSL 1.1.1e.

The RSA_get_multi_prime_extra_count(), RSA_get0_multi_prime_factors(), RSA_get0_multi_prime_crt_params(), RSA_set0_multi_prime_params(), and RSA_get_version() functions were added in OpenSSL 1.1.1.

Other functions described here were added in OpenSSL 1.1.0.

All of these functions were deprecated in OpenSSL 3.0.

## COPYRIGHT