NAME

BN_generate_prime_ex, BN_is_prime_ex, BN_is_prime_fasttest_ex, BN_GENCB_call, BN_GENCB_set_old, BN_GENCB_set, BN_generate_prime, BN_is_prime, BN_is_prime_fasttest - generate primes and test for primality

SYNOPSIS

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

int BN_generate_prime_ex(BIGNUM *ret,int bits,int safe, const BIGNUM *add,
const BIGNUM *rem, BN_GENCB *cb);

int BN_is_prime_ex(const BIGNUM *p,int nchecks, BN_CTX *ctx, BN_GENCB *cb);

int BN_is_prime_fasttest_ex(const BIGNUM *p,int nchecks, BN_CTX *ctx,
int do_trial_division, BN_GENCB *cb);

int BN_GENCB_call(BN_GENCB *cb, int a, int b);

#define BN_GENCB_set_old(gencb, callback, cb_arg) ...

#define BN_GENCB_set(gencb, callback, cb_arg) ...
```

Deprecated:

```
BIGNUM *BN_generate_prime(BIGNUM *ret, int num, int safe, BIGNUM *add,
BIGNUM *rem, void (*callback)(int, int, void *), void *cb_arg);

int BN_is_prime(const BIGNUM *a, int checks, void (*callback)(int, int,
void *), BN_CTX *ctx, void *cb_arg);

int BN_is_prime_fasttest(const BIGNUM *a, int checks,
void (*callback)(int, int, void *), BN_CTX *ctx, void *cb_arg,
int do_trial_division);
```

DESCRIPTION

BN_generate_prime_ex() generates a pseudo-random prime number of bit length **bits**. If**ret** is not **NULL**, it will be used to store the number.

If **cb** is not **NULL**, it is used as follows:

- **BN_GENCB_call(cb, 0, i)**is called after generating the i-th poten tial prime number.
- While the number is being tested for primality, **BN_GENCB_call(cb, 1, j)**is called as described below.
- When a prime has been found, **BN_GENCB_call(cb, 2, i)**is called.

The prime may have to fulfill additional requirements for use in Diffie-Hellman key exchange:

If **add** is not **NULL**, the prime will fulfill the condition p % **add** == **rem** (p % **add** == 1 if **rem** == **NULL**) in order to suit a given generator.

If **safe** is true, it will be a safe prime (i.e. a prime p so that (p-1)/2 is also prime).

The PRNG must be seeded prior to calling *BN_generate_prime_ex()*. The prime number generation has a negligible error probability.

BN_is_prime_ex() and *BN_is_prime_fasttest_ex()* test if the number **p** is prime. The following tests are performed until one of them shows that **p** is composite; if **p** passes all these tests, it is considered prime.

BN_is_prime_fasttest_ex(), when called with **do_trial_division == 1**, first attempts trial division by a number of small primes; if no divisors are found by this test and **cb** is not **NULL**,

BN_GENCB_call(cb, 1, -1)is called. If **do_trial_division == 0**, this test is skipped.

Both *BN_is_prime_ex()* and *BN_is_prime_fasttest_ex()* perform a Miller-Rabin probabilistic primality test with **nchecks** iterations. If **nchecks == BN_prime_checks**, a number of iterations is used that yields a false positive rate of at most 2^-80 for random input.

If **cb** is not **NULL**, **BN_GENCB_call(cb, 1, j)**is called after the j-th iteration (j = 0, 1, ...). **ctx** is a pre-allocated **BN_CTX** (to save the overhead of allocating and freeing the structure in a loop), or **NULL**.

BN_GENCB_call calls the callback function held in the **BN_GENCB** structure and passes the ints **a** and **b** as arguments. There are two types of **BN_GENCB** structure that are supported: "new" style and "old" style. New programs should prefer the "new" style, whilst the "old" style is provided for backwards compatibility purposes.

For "new" style callbacks a BN_GENCB structure should be initialised with a call to BN_GENCB_set, where **gencb** is a **BN_GENCB ***, **callback** is of type **int (*callback)(int, int, BN_GENCB *)**and **cb_arg** is a **v oid ***. "Old" style callbacks are the same except they are initialised with a call to BN_GENCB_set_old and **callback** is of type **void (*callback)(int, int, void *)**.

A callback is invoked through a call to **BN_GENCB_call**. This will check the type of the callback and will invoke **callback(a, b, gencb)**for new st yle callbacks or **callback(a, b, cb_arg)** for old style.

BN_generate_prime (deprecated) works in the same way as BN_generate_prime_ex but expects an old style callback function directly in the **callback** parameter, and an argument to pass to it in the **cb_arg**. Similarly BN_is_prime and BN_is_prime_fasttest are deprecated and can be compared to BN_is_prime_ex and BN_is_prime_fasttest_ex respectively.

RETURN VALUES

BN_generate_prime_ex() return 1 on success or 0 on error.

BN_is_prime_ex(), *BN_is_prime_fasttest_ex()*, *BN_is_prime()* and *BN_is_prime_fasttest()* return 0 if the number is composite, 1 if it is prime with an error probability of less than 0.25^**nchecks**, and -1 on error.

BN_generate_prime() returns the prime number on success, **NULL** otherwise.

Callback functions should return 1 on success or 0 on error.

The error codes can be obtained by *ERR_get_error(3)*.

SEE ALSO

bn(3), *ERR_get_error(3)*, *rand(3)*

HISTORY

The **cb_arg** arguments to *BN_generate_prime()* and to *BN_is_prime()* were added in SSLeay 0.9.0. The **ret** argument to *BN_generate_prime()* was added in SSLeay 0.9.1. *BN_is_prime_fasttest()* was added in OpenSSL 0.9.5.