

**NAME**

BN\_add, BN\_sub, BN\_mul, BN\_sqr, BN\_div, BN\_mod, BN\_nnmod, BN\_mod\_add, BN\_mod\_sub, BN\_mod\_mul, BN\_mod\_sqr, BN\_exp, BN\_mod\_exp, BN\_gcd - arithmetic operations on **BIGNUM**s

**SYNOPSIS**

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

int BN_add(BIGNUM *r, const BIGNUM *a, const BIGNUM *b);

int BN_sub(BIGNUM *r, const BIGNUM *a, const BIGNUM *b);

int BN_mul(BIGNUM *r, BIGNUM *a, BIGNUM *b, BN_CTX *ctx);

int BN_sqr(BIGNUM *r, BIGNUM *a, BN_CTX *ctx);

int BN_div(BIGNUM *dv, BIGNUM *rem, const BIGNUM *a, const BIGNUM *d,
BN_CTX *ctx);

int BN_mod(BIGNUM *rem, const BIGNUM *a, const BIGNUM *m, BN_CTX *ctx);

int BN_nnmod(BIGNUM *r, const BIGNUM *a, const BIGNUM *m, BN_CTX *ctx);

int BN_mod_add(BIGNUM *r, BIGNUM *a, BIGNUM *b, const BIGNUM *m,
BN_CTX *ctx);

int BN_mod_sub(BIGNUM *r, BIGNUM *a, BIGNUM *b, const BIGNUM *m,
BN_CTX *ctx);

int BN_mod_mul(BIGNUM *r, BIGNUM *a, BIGNUM *b, const BIGNUM *m,
BN_CTX *ctx);

int BN_mod_sqr(BIGNUM *r, BIGNUM *a, const BIGNUM *m, BN_CTX *ctx);

int BN_exp(BIGNUM *r, BIGNUM *a, BIGNUM *p, BN_CTX *ctx);

int BN_mod_exp(BIGNUM *r, BIGNUM *a, const BIGNUM *p,
const BIGNUM *m, BN_CTX *ctx);

int BN_gcd(BIGNUM *r, BIGNUM *a, BIGNUM *b, BN_CTX *ctx);
```

**DESCRIPTION**

*BN\_add()* adds *a* and *b* and places the result in *r* ( $r=a+b$ ). *r* may be the same **BIGNUM** as *a* or *b*.

*BN\_sub()* subtracts *b* from *a* and places the result in *r* ( $r=a-b$ ).

*BN\_mul()* multiplies *a* and *b* and places the result in *r* ( $r=a*b$ ). *r* may be the same **BIGNUM** as *a* or *b*. For multiplication by powers of 2, use [BN\\_lshift\(3\)](#).

*BN\_sqr()* takes the square of *a* and places the result in *r* ( $r=a^2$ ). *r* and *a* may be the same **BIGNUM**. This function is faster than *BN\_mul(r,a,a)*.

*BN\_div()* divides *a* by *d* and places the result in *dv* and the remainder in *rem* ( $dv=a/d$ ,  $rem=a\%d$ ). Either of *dv* and *rem* may be **NULL**, in which case the respective value is not returned. The result is rounded towards zero; thus if *a* is negative, the remainder will be zero or negative. For division by powers of 2, use [BN\\_rshift\(3\)](#).

*BN\_mod()* corresponds to *BN\_div()* with *dv* set to **NULL**.

*BN\_nnmod()* reduces *a* modulo *m* and places the non-negative remainder in *r*.

*BN\_mod\_add()* adds *a* to *b* modulo *m* and places the non-negative result in *r*.

*BN\_mod\_sub()* subtracts *b* from *a* modulo *m* and places the non-negative result in *r*.

*BN\_mod\_mul()* multiplies *a* by *b* and finds the non-negative remainder respective to modulus *m* ( $r=(a*b) \bmod m$ ). *r* may be the same **BIGNUM** as *a* or *b*. For more efficient algorithms for repeated computations using the same modulus, see [BN\\_mod\\_mul\\_montgomery\(3\)](#) and [BN\\_mod\\_mul\\_reciprocal\(3\)](#).

*BN\_mod\_sqr()* takes the square of *a* modulo **m** and places the result in *r*.

*BN\_exp()* raises *a* to the *p*-th power and places the result in *r* ( $r=ap$ ). This function is faster than repeated applications of *BN\_mul()*.

*BN\_mod\_exp()* computes *a* to the *p*-th power modulo *m* ( $r=ap \% m$ ). This function uses less time and space than *BN\_exp()*.

*BN\_gcd()* computes the greatest common divisor of *a* and *b* and places the result in *r*. *r* may be the same **BIGNUM** as *a* or *b*.

For all functions, *ctx* is a previously allocated **BN\_CTX** used for temporary variables; see [BN\\_CTX\\_new\(3\)](#).

Unless noted otherwise, the result **BIGNUM** must be different from the arguments.

## RETURN VALUES

For all functions, 1 is returned for success, 0 on error. The return value should always be checked (e.g., `if (!BN_add(r,a,b)) goto err;`). The error codes can be obtained by [ERR\\_get\\_error\(3\)](#).

## SEE ALSO

[bn\(3\)](#), [ERR\\_get\\_error\(3\)](#), [BN\\_CTX\\_new\(3\)](#), [BN\\_add\\_word\(3\)](#), [BN\\_set\\_bit\(3\)](#)

## HISTORY

*BN\_add()*, *BN\_sub()*, *BN\_sqr()*, *BN\_div()*, *BN\_mod()*, *BN\_mod\_mul()*, *BN\_mod\_exp()* and *BN\_gcd()* are available in all versions of SSLeay and OpenSSL. The *ctx* argument to *BN\_mul()* was added in SSLeay 0.9.1b. *BN\_exp()* appeared in SSLeay 0.9.0. *BN\_nnmod()*, *BN\_mod\_add()*, *BN\_mod\_sub()*, and *BN\_mod\_sqr()* were added in OpenSSL 0.9.7.