NAME

bn - multiprecision integer arithmetics

SYNOPSIS

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

BIGNUM *BN_new(void);
void BN_free(BIGNUM *a);
void BN_init(BIGNUM *);
void BN_clear(BIGNUM *a);
void BN_clear_free(BIGNUM *a);

BN_CTX *BN_CTX_new(void);
void BN_CTX_init(BN_CTX *c);
void BN_CTX_free(BN_CTX *c);

BIGNUM *BN_copy(BIGNUM *a, const BIGNUM *b);
BIGNUM *BN_dup(const BIGNUM *a);

BIGNUM *BN_swap(BIGNUM *a, BIGNUM *b);

int BN_num_bytes(const BIGNUM *a);
int BN_num_bits(const BIGNUM *a);
int BN_num_bits_word(BN_ULONG w);

void BN_set_negative(BIGNUM *a, int n);
int BN_is_negative(const BIGNUM *a);

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 *rem, const BIGNUM *a, const BIGNUM *m, BN_CTX *ctx);
int BN_mod_add(BIGNUM *ret, BIGNUM *a, BIGNUM *b, const BIGNUM *m,
BN_CTX *ctx);
int BN_mod_sub(BIGNUM *ret, BIGNUM *a, BIGNUM *b, const BIGNUM *m,
BN_CTX *ctx);
int BN_mod_mul(BIGNUM *ret, BIGNUM *a, BIGNUM *b, const BIGNUM *m,
BN_CTX *ctx);
int BN_mod_sqr(BIGNUM *ret, 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);

int BN_add_word(BIGNUM *a, BN_ULONG w);
int BN_sub_word(BIGNUM *a, BN_ULONG w);
int BN_mul_word(BIGNUM *a, BN_ULONG w);
BN_ULONG BN_div_word(BIGNUM *a, BN_ULONG w);
BN_ULONG BN_mod_word(const BIGNUM *a, BN_ULONG w);

int BN_cmp(BIGNUM *a, BIGNUM *b);
```

```
int BN_ucmp(BIGNUM *a, BIGNUM *b);
int BN_is_zero(BIGNUM *a);
int BN_is_one(BIGNUM *a);
int BN_is_word(BIGNUM *a, BN_ULONG w);
int BN_is_odd(BIGNUM *a);

int BN_zero(BIGNUM *a);
int BN_one(BIGNUM *a);
const BIGNUM *BN_value_one(void);
int BN_set_word(BIGNUM *a, unsigned long w);
unsigned long BN_get_word(BIGNUM *a);

int BN_rand(BIGNUM *rnd, int bits, int top, int bottom);
int BN_pseudo_rand(BIGNUM *rnd, int bits, int top, int bottom);
int BN_rand_range(BIGNUM *rnd, BIGNUM *range);
int BN_pseudo_rand_range(BIGNUM *rnd, BIGNUM *range);

BIGNUM *BN_generate_prime(BIGNUM *ret, int bits,int safe, BIGNUM *add,
BIGNUM *rem, void (*callback)(int, int, void *), void *cb_arg);
int BN_is_prime(const BIGNUM *p, int nchecks,
void (*callback)(int, int, void *), BN_CTX *ctx, void *cb_arg);

int BN_set_bit(BIGNUM *a, int n);
int BN_clear_bit(BIGNUM *a, int n);
int BN_is_bit_set(const BIGNUM *a, int n);
int BN_mask_bits(BIGNUM *a, int n);
int BN_lshift(BIGNUM *r, const BIGNUM *a, int n);
int BN_lshift1(BIGNUM *r, BIGNUM *a);
int BN_rshift(BIGNUM *r, BIGNUM *a, int n);
int BN_rshift1(BIGNUM *r, BIGNUM *a);

int BN_bn2bin(const BIGNUM *a, unsigned char *to);
BIGNUM *BN_bin2bn(const unsigned char *s, int len, BIGNUM *ret);
char *BN_bn2hex(const BIGNUM *a);
char *BN_bn2dec(const BIGNUM *a);
int BN_hex2bn(BIGNUM **a, const char *str);
int BN_dec2bn(BIGNUM **a, const char *str);
int BN_print(BIO *fp, const BIGNUM *a);
int BN_print_fp(FILE *fp, const BIGNUM *a);
int BN_bn2mpi(const BIGNUM *a, unsigned char *to);
BIGNUM *BN_mpi2bn(unsigned char *s, int len, BIGNUM *ret);

BIGNUM *BN_mod_inverse(BIGNUM *r, BIGNUM *a, const BIGNUM *n,
BN_CTX *ctx);

BN_RECP_CTX *BN_RECP_CTX_new(void);
void BN_RECP_CTX_init(BN_RECP_CTX *recp);
void BN_RECP_CTX_free(BN_RECP_CTX *recp);
int BN_RECP_CTX_set(BN_RECP_CTX *recp, const BIGNUM *m, BN_CTX *ctx);
int BN_mod_mul_reciprocal(BIGNUM *r, BIGNUM *a, BIGNUM *b,
BN_RECP_CTX *recp, BN_CTX *ctx);

BN_MONT_CTX *BN_MONT_CTX_new(void);
void BN_MONT_CTX_init(BN_MONT_CTX *ctx);
```

```
void BN_MONT_CTX_free(BN_MONT_CTX *mont);
int BN_MONT_CTX_set(BN_MONT_CTX *mont, const BIGNUM *m, BN_CTX *ctx);
BN_MONT_CTX *BN_MONT_CTX_copy(BN_MONT_CTX *to, BN_MONT_CTX *from);
int BN_mod_mul_montgomery(BIGNUM *r, BIGNUM *a, BIGNUM *b,
BN_MONT_CTX *mont, BN_CTX *ctx);
int BN_from_montgomery(BIGNUM *r, BIGNUM *a, BN_MONT_CTX *mont,
BN_CTX *ctx);
int BN_to_montgomery(BIGNUM *r, BIGNUM *a, BN_MONT_CTX *mont,
BN_CTX *ctx);

BN_BLINDING *BN_BLINDING_new(const BIGNUM *A, const BIGNUM *Ai,
BIGNUM *mod);
void BN_BLINDING_free(BN_BLINDING *b);
int BN_BLINDING_update(BN_BLINDING *b,BN_CTX *ctx);
int BN_BLINDING_convert(BIGNUM *n, BN_BLINDING *b, BN_CTX *ctx);
int BN_BLINDING_invert(BIGNUM *n, BN_BLINDING *b, BN_CTX *ctx);
int BN_BLINDING_convert_ex(BIGNUM *n, BIGNUM *r, BN_BLINDING *b,
BN_CTX *ctx);
int BN_BLINDING_invert_ex(BIGNUM *n,const BIGNUM *r,BN_BLINDING *b,
BN_CTX *ctx);
unsigned long BN_BLINDING_get_thread_id(const BN_BLINDING *);
void BN_BLINDING_set_thread_id(BN_BLINDING *, unsigned long);
unsigned long BN_BLINDING_get_flags(const BN_BLINDING *);
void BN_BLINDING_set_flags(BN_BLINDING *, unsigned long);
BN_BLINDING *BN_BLINDING_create_param(BN_BLINDING *b,
const BIGNUM *e, BIGNUM *m, BN_CTX *ctx,
int (*bn_mod_exp)(BIGNUM *r, const BIGNUM *a, const BIGNUM *p,
const BIGNUM *m, BN_CTX *ctx, BN_MONT_CTX *m_ctx),
BN_MONT_CTX *m_ctx);
```

DESCRIPTION

This library performs arithmetic operations on integers of arbitrary size. It was written for use in public key cryptography, such as RSA and Diffie-Hellman.

It uses dynamic memory allocation for storing its data structures. That means that there is no limit on the size of the numbers manipulated by these functions, but return values must always be checked in case a memory allocation error has occurred.

The basic object in this library is a **BIGNUM**. It is used to hold a single large integer. This type should be considered opaque and fields should not be modified or accessed directly.

The creation of **BIGNUM** objects is described in *BN_new(3)*; *BN_add(3)* describes most of the arithmetic operations. Comparison is described in *BN_cmp(3)*; *BN_zero(3)* describes certain assignments, *BN_rand(3)* the generation of random numbers, *BN_generate_prime(3)* deals with prime numbers and *BN_set_bit(3)* with bit operations. The conversion of **BIGNUM**s to external formats is described in *BN_bn2bin(3)*.

SEE ALSO

bn_internal(3), *dh(3)*, *err(3)*, *rand(3)*, *rsa(3)*, *BN_new(3)*, *BN_CTX_new(3)*, *BN_copy(3)*, *BN_swap(3)*, *BN_num_bytes(3)*, *BN_add(3)*, *BN_add_word(3)*, *BN_cmp(3)*, *BN_zero(3)*, *BN_rand(3)*, *BN_generate_prime(3)*, *BN_set_bit(3)*, *BN_bn2bin(3)*, *BN_mod_inverse(3)*, *BN_mod_mul_reciprocal(3)*, *BN_mod_mul_montgomery(3)*, *BN_BLINDING_new(3)*