NAME

BN_mod_mul_reciprocal, BN_div_recp, BN_RECP_CTX_new, BN_RECP_CTX_free, BN_RECP_CTX_set - modular multiplication using reciprocal

SYNOPSIS

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

BN_RECP_CTX *BN_RECP_CTX_new(void);
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_div_recp(BIGNUM *dv, BIGNUM *rem, const BIGNUM *a, BN_RECP_CTX *recp,
                BN_CTX *ctx);

int BN_mod_mul_reciprocal(BIGNUM *r, const BIGNUM *a, const BIGNUM *b,
                          BN_RECP_CTX *recp, BN_CTX *ctx);
```

DESCRIPTION

BN_mod_mul_reciprocal() can be used to perform an efficient **BN_mod_mul**(3) operation when the operation will be performed repeatedly with the same modulus. It computes **r**=(**a***__b__)%**m** using **recp**=1/**m**, which is set as described below. **ctx** is a previously allocated **BN_CTX** used for temporary variables.

BN_RECP_CTX_new() allocates and initializes a **BN_RECP** structure.

BN_RECP_CTX_free() frees the components of the **BN_RECP**, and, if it was created by **BN_RECP_CTX_new()**, also the structure itself. If **recp** is NULL, nothing is done.

BN_RECP_CTX_set() stores **m** in **recp** and sets it up for computing 1/**m** and shifting it left by BN_num_bits(**m**)+1 to make it an integer. The result and the number of bits it was shifted left will later be stored in **recp**.

BN_div_recp() divides **a** by **m** using **recp**. It places the quotient in **dv** and the remainder in **rem**.

The **BN_RECP_CTX** structure cannot be shared between threads.

RETURN VALUES

BN_RECP_CTX_new() returns the newly allocated **BN_RECP_CTX**, and NULL on error.

BN_RECP_CTX_free() has no return value.

For the other functions, 1 is returned for success, 0 on error. The error codes can be obtained by **ERR_get_error**(3).

SEE ALSO

ERR_get_error(3), **BN_add**(3), **BN_CTX_new**(3)

HISTORY

BN_RECP_CTX_init() was removed in OpenSSL 1.1.0

COPYRIGHT