

**NAME**

BN\_BLINDING\_new, BN\_BLINDING\_free, BN\_BLINDING\_update, BN\_BLINDING\_convert, BN\_BLINDING\_invert, BN\_BLINDING\_convert\_ex, BN\_BLINDING\_invert\_ex, BN\_BLINDING\_is\_current\_thread, BN\_BLINDING\_set\_current\_thread, BN\_BLINDING\_lock, BN\_BLINDING\_unlock, BN\_BLINDING\_get\_flags, BN\_BLINDING\_set\_flags, BN\_BLINDING\_create\_param - blinding related BIGNUM functions

**SYNOPSIS**

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

```
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);
int BN_BLINDING_is_current_thread(BN_BLINDING *b);
void BN_BLINDING_set_current_thread(BN_BLINDING *b);
int BN_BLINDING_lock(BN_BLINDING *b);
int BN_BLINDING_unlock(BN_BLINDING *b);
unsigned long BN_BLINDING_get_flags(const BN_BLINDING *b);
void BN_BLINDING_set_flags(BN_BLINDING *b, unsigned long flags);
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**

**BN\_BLINDING\_new()** allocates a new **BN\_BLINDING** structure and copies the **A** and **Ai** values into the newly created **BN\_BLINDING** object.

**BN\_BLINDING\_free()** frees the **BN\_BLINDING** structure. If **b** is NULL, nothing is done.

**BN\_BLINDING\_update()** updates the **BN\_BLINDING** parameters by squaring the **A** and **Ai** or, after specific number of uses and if the necessary parameters are set, by re-creating the blinding parameters.

**BN\_BLINDING\_convert\_ex()** multiplies **n** with the blinding factor **A**. If **r** is not NULL a copy the inverse blinding factor **Ai** will be returned in **r** (this is useful if a **RSA** object is shared among several threads). **BN\_BLINDING\_invert\_ex()** multiplies **n** with the inverse blinding factor **Ai**. If **r** is not NULL it will be used as the inverse blinding.

**BN\_BLINDING\_convert()** and **BN\_BLINDING\_invert()** are wrapper functions for **BN\_BLINDING\_convert\_ex()** and **BN\_BLINDING\_invert\_ex()** with **r** set to NULL.

**BN\_BLINDING\_is\_current\_thread()** returns whether the **BN\_BLINDING** structure is owned by the current thread. This is to help users provide proper locking if needed for multi-threaded use.

**BN\_BLINDING\_set\_current\_thread()** sets the current thread as the owner of the **BN\_BLINDING** structure.

**BN\_BLINDING\_lock()** locks the **BN\_BLINDING** structure.

**BN\_BLINDING\_unlock()** unlocks the **BN\_BLINDING** structure.

**BN\_BLINDING\_get\_flags()** returns the **BN\_BLINDING** flags. Currently there are two supported flags: **BN\_BLINDING\_NO\_UPDATE** and **BN\_BLINDING\_NO\_RECREATE**.

**BN\_BLINDING\_NO\_UPDATE** inhibits the automatic update of the **BN\_BLINDING** parameters after each use and **BN\_BLINDING\_NO\_RECREATE** inhibits the automatic re-creation of the **BN\_BLINDING** parameters after a fixed number of uses (currently 32). In newly allocated **BN\_BLINDING** objects no flags are set. **BN\_BLINDING\_set\_flags()** sets the **BN\_BLINDING** parameters flags.

**BN\_BLINDING\_create\_param()** creates new **BN\_BLINDING** parameters using the exponent **e** and the modulus **m**. **bn\_mod\_exp** and **m\_ctx** can be used to pass special functions for exponentiation (normally **BN\_mod\_exp\_mont()** and **BN\_MONT\_CTX**).

## RETURN VALUES

**BN\_BLINDING\_new()** returns the newly allocated **BN\_BLINDING** structure or NULL in case of an error.

**BN\_BLINDING\_update()**, **BN\_BLINDING\_convert()**, **BN\_BLINDING\_invert()**,

**BN\_BLINDING\_convert\_ex()** and **BN\_BLINDING\_invert\_ex()** return 1 on success and 0 if an error occurred.

**BN\_BLINDING\_is\_current\_thread()** returns 1 if the current thread owns the **BN\_BLINDING** object, 0 otherwise.

**BN\_BLINDING\_set\_current\_thread()** doesn't return anything.

**BN\_BLINDING\_lock()**, **BN\_BLINDING\_unlock()** return 1 if the operation succeeded or 0 on error.

**BN\_BLINDING\_get\_flags()** returns the currently set **BN\_BLINDING** flags (a **unsigned long** value).

**BN\_BLINDING\_create\_param()** returns the newly created **BN\_BLINDING** parameters or NULL on error.

## HISTORY

**BN\_BLINDING\_thread\_id()** was first introduced in OpenSSL 1.0.0, and it deprecates **BN\_BLINDING\_set\_thread\_id()** and **BN\_BLINDING\_get\_thread\_id()**.

## COPYRIGHT

Copyright 2005-2021 The OpenSSL Project Authors. All Rights Reserved.

Licensed under the Apache License 2.0 (the "License"). You may not use this file except in compliance with the License. You can obtain a copy in the file LICENSE in the source distribution or at <<https://www.openssl.org/source/license.html>>.