## NAME

EC_POINT_add, EC_POINT_dbl, EC_POINT_invert, EC_POINT_is_at_infinity, EC_POINT_is_on_curve, EC_POINT_cmp, EC_POINT_make_affine, EC_POINTs_make_affine, EC_POINTs_mul, EC_POINT_mul, EC_GROUP_precompute_mult, EC_GROUP_have_precompute_mult - Functions for performing mathematical operations and tests on EC_POINT objects

## SYNOPSIS

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

int EC_POINT_add(const EC_GROUP *group, EC_POINT *r, const EC_POINT *a,
                 const EC_POINT *b, BN_CTX *ctx);
int EC_POINT_dbl(const EC_GROUP *group, EC_POINT *r, const EC_POINT *a, BN_CTX *ctx);
int EC_POINT_invert(const EC_GROUP *group, EC_POINT *a, BN_CTX *ctx);
int EC_POINT_is_at_infinity(const EC_GROUP *group, const EC_POINT *p);
int EC_POINT_is_on_curve(const EC_GROUP *group, const EC_POINT *point, BN_CTX *ctx);
int EC_POINT_cmp(const EC_GROUP *group, const EC_POINT *a, const EC_POINT *b, BN_CTX *ctx);
int EC_POINT_mul(const EC_GROUP *group, EC_POINT *r, const BIGNUM *n,
                 const EC_POINT *q, const BIGNUM *m, BN_CTX *ctx);
```

The following functions have been deprecated since OpenSSL 3.0, and can be hidden entirely by defining **OPENSSL_API_COMPAT** with a suitable version value, see **openssl_user_macros**(7):

```
int EC_POINT_make_affine(const EC_GROUP *group, EC_POINT *point, BN_CTX *ctx);
int EC_POINTs_make_affine(const EC_GROUP *group, size_t num,
                          EC_POINT *points[], BN_CTX *ctx);
int EC_POINTs_mul(const EC_GROUP *group, EC_POINT *r, const BIGNUM *n, size_t num,
                  const EC_POINT *p[], const BIGNUM *m[], BN_CTX *ctx);
int EC_GROUP_precompute_mult(EC_GROUP *group, BN_CTX *ctx);
int EC_GROUP_have_precompute_mult(const EC_GROUP *group);
```

## DESCRIPTION

EC_POINT_add adds the two points **a** and **b** and places the result in **r**. Similarly EC_POINT_dbl doubles the point **a** and places the result in **r**. In both cases it is valid for **r** to be one of **a** or **b**.

EC_POINT_invert calculates the inverse of the supplied point **a**. The result is placed back in **a**.

The function EC_POINT_is_at_infinity tests whether the supplied point is at infinity or not.

EC_POINT_is_on_curve tests whether the supplied point is on the curve or not.

EC_POINT_cmp compares the two supplied points and tests whether or not they are equal.

The functions EC_POINT_make_affine and EC_POINTs_make_affine force the internal representation of the EC_POINT(s) into the affine coordinate system. In the case of EC_POINTs_make_affine the value **num** provides the number of points in the array **points** to be forced. These functions were deprecated in OpenSSL 3.0 and should no longer be used. Modern versions automatically perform this conversion when needed.

EC_POINT_mul calculates the value generator * **n** + **q** * **m** and stores the result in **r**. The value **n** may be NULL in which case the result is just **q** * **m** (variable point multiplication). Alternatively, both **q** and **m** may be NULL, and **n** non-NULL, in which case the result is just generator * **n** (fixed point multiplication). When performing a single fixed or variable point multiplication, the underlying implementation uses a constant time algorithm, when the input scalar (either **n** or **m**) is in the range [0, ec_group_order).

Although deprecated in OpenSSL 3.0 and should no longer be used, EC_POINTs_mul calculates the value generator * **n** + **q[0]** * **m[0]** + ... + **q[num-1]** * **m[num-1]**. As for EC_POINT_mul the value **n** may be NULL or **num** may be zero. When performing a fixed point multiplication (**n** is non-NULL and **num** is 0) or a variable point multiplication (**n** is NULL and **num** is 1), the underlying implementation uses a constant time algorithm, when the input scalar (either **n** or **m[0]**) is in the range [0, ec_group_order). Modern versions should instead use **EC_POINT_mul()**, combined (if needed) with **EC_POINT_add()** in such rare circumstances.

The function EC_GROUP_precompute_mult stores multiples of the generator for faster point multiplication, whilst EC_GROUP_have_precompute_mult tests whether precomputation has already been done. See **EC_GROUP_copy**(3) for information about the generator. Precomputation functionality was deprecated in OpenSSL 3.0. Users of **EC_GROUP_precompute_mult()** and **EC_GROUP_have_precompute_mult()** should switch to named curves which OpenSSL has hardcoded lookup tables for.

## RETURN VALUES

The following functions return 1 on success or 0 on error: EC_POINT_add, EC_POINT_dbl, EC_POINT_invert, EC_POINT_make_affine, EC_POINTs_make_affine, EC_POINTs_make_affine, EC_POINT_mul, EC_POINTs_mul and EC_GROUP_precompute_mult.

EC_POINT_is_at_infinity returns 1 if the point is at infinity, or 0 otherwise.

EC_POINT_is_on_curve returns 1 if the point is on the curve, 0 if not, or -1 on error.

EC_POINT_cmp returns 1 if the points are not equal, 0 if they are, or -1 on error.

EC_GROUP_have_precompute_mult return 1 if a precomputation has been done, or 0 if not.

## SEE ALSO

**crypto**(7), **EC_GROUP_new**(3), **EC_GROUP_copy**(3), **EC_POINT_new**(3), **EC_KEY_new**(3), **EC_GFp_simple_method**(3), **d2i_ECPKParameters**(3)

## HISTORY

**EC_POINT_make_affine()**, **EC_POINTs_make_affine()**, **EC_POINTs_mul()**, **EC_GROUP_precompute_mult()**, and **EC_GROUP_have_precompute_mult()** were deprecated in OpenSSL 3.0.

## COPYRIGHT

Copyright 2013-2023 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>.