

**NAME**

EC\_GROUP\_new, EC\_GROUP\_free, EC\_GROUP\_clear\_free, EC\_GROUP\_new\_curve\_GFp, EC\_GROUP\_new\_curve\_GF2m, EC\_GROUP\_new\_by\_curve\_name, EC\_GROUP\_set\_curve\_GFp, EC\_GROUP\_get\_curve\_GFp, EC\_GROUP\_set\_curve\_GF2m, EC\_GROUP\_get\_curve\_GF2m, EC\_get\_builtin\_curves - Functions for creating and destroying EC\_GROUP objects.

**SYNOPSIS**

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

```
EC_GROUP *EC_GROUP_new(const EC_METHOD *meth);
void EC_GROUP_free(EC_GROUP *group);
void EC_GROUP_clear_free(EC_GROUP *group);
```

```
EC_GROUP *EC_GROUP_new_curve_GFp(const BIGNUM *p, const BIGNUM *a, const BIGNUM *b, BN_CTX *c);
EC_GROUP *EC_GROUP_new_curve_GF2m(const BIGNUM *p, const BIGNUM *a, const BIGNUM *b, BN_CTX *c);
EC_GROUP *EC_GROUP_new_by_curve_name(int nid);
```

```
int EC_GROUP_set_curve_GFp(EC_GROUP *group, const BIGNUM *p, const BIGNUM *a, const BIGNUM *b, BN_CTX *c);
int EC_GROUP_get_curve_GFp(const EC_GROUP *group, BIGNUM *p, BIGNUM *a, BIGNUM *b, BN_CTX *c);
int EC_GROUP_set_curve_GF2m(EC_GROUP *group, const BIGNUM *p, const BIGNUM *a, const BIGNUM *b, BN_CTX *c);
int EC_GROUP_get_curve_GF2m(const EC_GROUP *group, BIGNUM *p, BIGNUM *a, BIGNUM *b, BN_CTX *c);
```

```
size_t EC_get_builtin_curves(EC_builtin_curve *r, size_t nitems);
```

**DESCRIPTION**

Within the library there are two forms of elliptic curve that are of interest. The first form is those defined over the prime field  $F_p$ . The elements of  $F_p$  are the integers 0 to  $p-1$ , where  $p$  is a prime number. This gives us a revised elliptic curve equation as follows:

$$y^2 \bmod p = x^3 + ax + b \bmod p$$

The second form is those defined over a binary field  $F_2^m$  where the elements of the field are integers of length at most  $m$  bits. For this form the elliptic curve equation is modified to:

$$y^2 + xy = x^3 + ax^2 + b \text{ (where } b \neq 0 \text{)}$$

Operations in a binary field are performed relative to an **irreducible polynomial**. All such curves with OpenSSL use a trinomial or a pentanomial for this parameter.

A new curve can be constructed by calling EC\_GROUP\_new, using the implementation provided by **meth** (see [EC\\_GFp\\_simple\\_method\(3\)](#)). It is then necessary to call either EC\_GROUP\_set\_curve\_GFp or EC\_GROUP\_set\_curve\_GF2m as appropriate to create a curve defined over  $F_p$  or over  $F_2^m$  respectively.

EC\_GROUP\_set\_curve\_GFp sets the curve parameters **p**, **a** and **b** for a curve over  $F_p$  stored in **group**. EC\_group\_get\_curve\_GFp obtains the previously set curve parameters.

EC\_GROUP\_set\_curve\_GF2m sets the equivalent curve parameters for a curve over  $F_2^m$ . In this case **p** represents the irreducible polynomial - each bit represents a term in the polynomial. Therefore there will either be three or five bits set dependant on whether the polynomial is a trinomial or a pentanomial. EC\_group\_get\_curve\_GF2m obtains the previously set curve parameters.

The functions EC\_GROUP\_new\_curve\_GFp and EC\_GROUP\_new\_curve\_GF2m are shortcuts for calling EC\_GROUP\_new and the appropriate EC\_group\_set\_curve function. An appropriate default implementation method will be used.

Whilst the library can be used to create any curve using the functions described above, there are also a number of predefined curves that are available. In order to obtain a list of all of the predefined curves, call the function EC\_get\_builtin\_curves. The parameter **r** should be an array of

EC\_builtin\_curve structures of size **nitems**. The function will populate the **r** array with information about the builtin curves. If **nitems** is less than the total number of curves available, then the first **nitems** curves will be returned. Otherwise the total number of curves will be provided. The return value is the total number of curves available (whether that number has been populated in **r** or not). Passing a NULL **r**, or setting **nitems** to 0 will do nothing other than return the total number of curves available. The EC\_builtin\_curve structure is defined as follows:

```
typedef struct {
    int nid;
    const char *comment;
} EC_builtin_curve;
```

Each EC\_builtin\_curve item has a unique integer id (**nid**), and a human readable comment string describing the curve.

In order to construct a builtin curve use the function EC\_GROUP\_new\_by\_curve\_name and provide the **nid** of the curve to be constructed.

EC\_GROUP\_free frees the memory associated with the EC\_GROUP.

EC\_GROUP\_clear\_free destroys any sensitive data held within the EC\_GROUP and then frees its memory.

## RETURN VALUES

All EC\_GROUP\_new\* functions return a pointer to the newly constructed group, or NULL on error.

EC\_get\_builtin\_curves returns the number of builtin curves that are available.

EC\_GROUP\_set\_curve\_GFp, EC\_GROUP\_get\_curve\_GFp, EC\_GROUP\_set\_curve\_GF2m, EC\_GROUP\_get\_curve\_GF2m return 1 on success or 0 on error.

## SEE ALSO

[crypto\(3\)](#), [ec\(3\)](#), [EC\\_GROUP\\_copy\(3\)](#), [EC\\_POINT\\_new\(3\)](#), [EC\\_POINT\\_add\(3\)](#), [EC\\_KEY\\_new\(3\)](#), [EC\\_GFp\\_simple\\_method\(3\)](#), [d2i\\_ECPKParameters\(3\)](#)