NAME

PKCS5_PBE_keyivgen, PKCS5_PBE_keyivgen_ex, PKCS5_pbe2_set, PKCS5_pbe2_set_iv, PKCS5_pbe2_set_iv_ex, PKCS5_pbe_set, PKCS5_pbe_set_ex, PKCS5_pbe2_set_scrypt, PKCS5_pbe_set0_algor, PKCS5_pbe_set0_algor_ex, PKCS5_v2_PBE_keyivgen, PKCS5_v2_PBE_keyivgen_ex, PKCS5_v2_scrypt_keyivgen, PKCS5_v2_scrypt_keyivgen_ex, PKCS5_pbkdf2_set, PKCS5_pbkdf2_set_ex, EVP_PBE_scrypt, EVP_PBE_scrypt_ex - PKCS#5 Password based encryption routines

SYNOPSIS

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

int PKCS5_PBE_keyivgen(EVP_CIPHER_CTX *ctx, const char *pass, int passlen,
                       ASN1_TYPE *param, const EVP_CIPHER *cipher,
                       const EVP_MD *md, int en_de);
int PKCS5_PBE_keyivgen_ex(EVP_CIPHER_CTX *cctx, const char *pass, int passlen,
                          ASN1_TYPE *param, const EVP_CIPHER *cipher,
                          const EVP_MD *md, int en_de, OSSL_LIB_CTX *libctx,
                          const char *propq);
int PKCS5_v2_PBE_keyivgen(EVP_CIPHER_CTX *ctx, const char *pass, int passlen,
                          ASN1_TYPE *param, const EVP_CIPHER *cipher,
                          const EVP_MD *md, int en_de);
int PKCS5_v2_PBE_keyivgen_ex(EVP_CIPHER_CTX *ctx, const char *pass, int passlen,
                             ASN1_TYPE *param, const EVP_CIPHER *cipher,
                             const EVP_MD *md, int en_de,
                             OSSL_LIB_CTX *libctx, const char *propq);
int EVP_PBE_scrypt(const char *pass, size_t passlen,
                   const unsigned char *salt, size_t saltlen,
                   uint64_t N, uint64_t r, uint64_t p, uint64_t maxmem,
                   unsigned char *key, size_t keylen);
int EVP_PBE_scrypt_ex(const char *pass, size_t passlen,
                      const unsigned char *salt, size_t saltlen,
                      uint64_t N, uint64_t r, uint64_t p, uint64_t maxmem,
                      unsigned char *key, size_t keylen,
                      OSSL_LIB_CTX *ctx, const char *propq);
int PKCS5_v2_scrypt_keyivgen(EVP_CIPHER_CTX *ctx, const char *pass,
                             int passlen, ASN1_TYPE *param,
                             const EVP_CIPHER *c, const EVP_MD *md, int en_de);
int PKCS5_v2_scrypt_keyivgen_ex(EVP_CIPHER_CTX *ctx, const char *pass,
                                int passlen, ASN1_TYPE *param,
                                const EVP_CIPHER *c, const EVP_MD *md, int en_de,
```

```
                        OSSL_LIB_CTX *libctx, const char *propq);

    #include <openssl/x509.h>

    int PKCS5_pbe_set0_algor(X509_ALGOR *algor, int alg, int iter,
                             const unsigned char *salt, int saltlen);
    int PKCS5_pbe_set0_algor_ex(X509_ALGOR *algor, int alg, int iter,
                                const unsigned char *salt, int saltlen,
                                OSSL_LIB_CTX *libctx);

    X509_ALGOR *PKCS5_pbe_set(int alg, int iter,
                              const unsigned char *salt, int saltlen);
    X509_ALGOR *PKCS5_pbe_set_ex(int alg, int iter,
                                 const unsigned char *salt, int saltlen,
                                 OSSL_LIB_CTX *libctx);

    X509_ALGOR *PKCS5_pbe2_set(const EVP_CIPHER *cipher, int iter,
                               unsigned char *salt, int saltlen);
    X509_ALGOR *PKCS5_pbe2_set_iv(const EVP_CIPHER *cipher, int iter,
                                  unsigned char *salt, int saltlen,
                                  unsigned char *aiv, int prf_nid);
    X509_ALGOR *PKCS5_pbe2_set_iv_ex(const EVP_CIPHER *cipher, int iter,
                                     unsigned char *salt, int saltlen,
                                     unsigned char *aiv, int prf_nid,
                                     OSSL_LIB_CTX *libctx);
    X509_ALGOR *PKCS5_pbe2_set_scrypt(const EVP_CIPHER *cipher,
                                      const unsigned char *salt, int saltlen,
                                      unsigned char *aiv, uint64_t N, uint64_t r,
                                      uint64_t p);

    X509_ALGOR *PKCS5_pbkdf2_set(int iter, unsigned char *salt, int saltlen,
                                 int prf_nid, int keylen);
    X509_ALGOR *PKCS5_pbkdf2_set_ex(int iter, unsigned char *salt, int saltlen,
                                    int prf_nid, int keylen,
                                    OSSL_LIB_CTX *libctx);
```

DESCRIPTION

Key Derivation

PKCS5_PBE_keyivgen() and **PKCS5_PBE_keyivgen_ex()** take a password *pass* of length *passlen*, parameters *param* and a message digest function *md_type* and performs a key derivation according to

PKCS#5 PBES1. The resulting key is then used to initialise the cipher context *ctx* with a cipher *cipher* for encryption (*en_de*=1) or decryption (*en_de*=0).

pass is an optional parameter and can be NULL. If *passlen* is -1, then the function will calculate the length of *pass* using **strlen()**.

PKCS5_v2_PBE_keyivgen() and **PKCS5_v2_PBE_keyivgen_ex()** are similar to the above but instead use PKCS#5 PBES2 as the encryption algorithm using the supplied parameters.

PKCS5_v2_scrypt_keyivgen() and **PKCS5_v2_scrypt_keyivgen_ex()** use SCRYPT as the key derivation part of the encryption algorithm.

salt is the salt used in the derivation of length *saltlen*. If the *salt* is NULL, then *saltlen* must be 0. The function will not attempt to calculate the length of the *salt* because it is not assumed to be NULL terminated.

iter is the iteration count and its value should be greater than or equal to 1. RFC 2898 suggests an iteration count of at least 1000. Any *iter* less than 1 is treated as a single iteration.

digest is the message digest function used in the derivation.

Functions ending in **_ex()** take optional parameters *libctx* and *propq* which are used to select appropriate algorithm implementations.

Algorithm Identifier Creation

PKCS5_pbe_set(), **PKCS5_pbe_set_ex()**, **PKCS5_pbe2_set()**, **PKCS5_pbe2_set_iv()**, **PKCS5_pbe2_set_iv_ex()** and **PKCS5_pbe2_set_scrypt()** generate an **X509_ALGOR** object which represents an AlgorithmIdentifier containing the algorithm OID and associated parameters for the PBE algorithm.

PKCS5_pbkdf2_set() and **PKCS5_pbkdf2_set_ex()** generate an **X509_ALGOR** object which represents an AlgorithmIdentifier containing the algorithm OID and associated parameters for the PBKDF2 algorithm.

PKCS5_pbe_set0_algor() and **PKCS5_pbe_set0_algor_ex()** set the PBE algorithm OID and parameters into the supplied **X509_ALGOR**.

NOTES

The ***_keyivgen()** functions are typically used in PKCS#12 to encrypt objects.

These functions make no assumption regarding the given password. It will simply be treated as a byte sequence.

RETURN VALUES

PKCS5_PBE_keyivgen(), **PKCS5_v2_PBE_keyivgen()**, **PKCS5_v2_PBE_keyivgen_ex()**, **PKCS5_v2_scrypt_keyivgen()**, **PKCS5_v2_scrypt_keyivgen_ex()**, **PKCS5_pbe_set0_algor()** and **PKCS5_pbe_set0_algor_ex()** return 1 for success and 0 if an error occurs.

PKCS5_pbe_set(), **PKCS5_pbe_set_ex()**, **PKCS5_pbe2_set()**, **PKCS5_pbe2_set_iv()**, **PKCS5_pbe2_set_iv_ex()**, **PKCS5_pbe2_set_scrypt()**, **PKCS5_pbkdf2_set()** and **PKCS5_pbkdf2_set_ex()** return an **X509_ALGOR** object or NULL if an error occurs.

CONFORMING TO

IETF RFC 8018 (<https://tools.ietf.org/html/rfc8018>)

SEE ALSO

EVP_PBE_CipherInit_ex(3), **PKCS12_pbe_crypt_ex**(3), **passphrase-encoding**(7)

HISTORY

PKCS5_v2_PBE_keyivgen_ex(), **EVP_PBE_scrypt_ex()**, **PKCS5_v2_scrypt_keyivgen_ex()**, **PKCS5_pbe_set0_algor_ex()**, **PKCS5_pbe_set_ex()**, **PKCS5_pbe2_set_iv_ex()** and **PKCS5_pbkdf2_set_ex()** were added in OpenSSL 3.0.

From OpenSSL 3.0 the PBKDF1 algorithm used in **PKCS5_PBE_keyivgen()** and **PKCS5_PBE_keyivgen_ex()** has been moved to the legacy provider as an EVP_KDF.

COPYRIGHT