NAME

EVP_PKEY_encapsulate_init, EVP_PKEY_encapsulate - Key encapsulation using a KEM algorithm with a public key

SYNOPSIS

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

 int EVP_PKEY_encapsulate_init(EVP_PKEY_CTX *ctx, const OSSL_PARAM params[]);
 int EVP_PKEY_encapsulate(EVP_PKEY_CTX *ctx,
                          unsigned char *wrappedkey, size_t *wrappedkeylen,
                          unsigned char *genkey, size_t *genkeylen);
```

DESCRIPTION

The **EVP_PKEY_encapsulate_init()** function initializes a public key algorithm context *ctx* for an encapsulation operation and then sets the *params* on the context in the same way as calling **EVP_PKEY_CTX_set_params**(3). Note that *ctx* is usually is produced using **EVP_PKEY_CTX_new_from_pkey**(3), specifying the public key to use.

The **EVP_PKEY_encapsulate()** function performs a public key encapsulation operation using *ctx*. The symmetric secret generated in *genkey* can be used as key material. The ciphertext in *wrappedkey* is its encapsulated form, which can be sent to another party, who can use **EVP_PKEY_decapsulate**(3) to retrieve it using their private key. If *wrappedkey* is NULL then the maximum size of the output buffer is written to the **wrappedkeylen* parameter unless *wrappedkeylen* is NULL and the maximum size of the generated key buffer is written to **genkeylen* unless *genkeylen* is NULL. If *wrappedkey* is not NULL and the call is successful then the internally generated key is written to *genkey* and its size is written to **genkeylen*. The encapsulated version of the generated key is written to *wrappedkey* and its size is written to **wrappedkeylen*.

NOTES

After the call to **EVP_PKEY_encapsulate_init()** algorithm-specific parameters for the operation may be set or modified using **EVP_PKEY_CTX_set_params**(3).

RETURN VALUES

EVP_PKEY_encapsulate_init() and **EVP_PKEY_encapsulate()** return 1 for success and 0 or a negative value for failure. In particular a return value of -2 indicates the operation is not supported by the public key algorithm.

EXAMPLES

Encapsulate an RSASVE key (for RSA keys).

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

/*
 * NB: assumes rsa_pub_key is an public key of another party.
 */

EVP_PKEY_CTX *ctx = NULL;
size_t secretlen = 0, outlen = 0;
unsigned char *out = NULL, *secret = NULL;

ctx = EVP_PKEY_CTX_new_from_pkey(libctx, rsa_pub_key, NULL);
if (ctx = NULL)
    /* Error */
if (EVP_PKEY_encapsulate_init(ctx, NULL) <= 0)
    /* Error */

/* Set the mode - only ’RSASVE’ is currently supported */
 if (EVP_PKEY_CTX_set_kem_op(ctx, "RSASVE") <= 0)
    /* Error */
/* Determine buffer length */
if (EVP_PKEY_encapsulate(ctx, NULL, &outlen, NULL, &secretlen) <= 0)
    /* Error */

out = OPENSSL_malloc(outlen);
secret = OPENSSL_malloc(secretlen);
if (out == NULL || secret == NULL)
    /* malloc failure */

/*
 * The generated ’secret’ can be used as key material.
 * The encapsulated ’out’ can be sent to another party who can
 * decapsulate it using their private key to retrieve the ’secret’.
 */
if (EVP_PKEY_encapsulate(ctx, out, &outlen, secret, &secretlen) <= 0)
    /* Error */
```

SEE ALSO

EVP_PKEY_CTX_new_from_pkey(3), **EVP_PKEY_decapsulate**(3), **EVP_KEM-RSA**(7),

HISTORY

These functions were added in OpenSSL 3.0.

COPYRIGHT