NAME

EVP_PKEY_verify_recover_init, EVP_PKEY_verify_recover_init_ex, EVP_PKEY_verify_recover - recover signature using a public key algorithm

SYNOPSIS

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

int EVP_PKEY_verify_recover_init(EVP_PKEY_CTX *ctx);
int EVP_PKEY_verify_recover_init_ex(EVP_PKEY_CTX *ctx,
                                    const OSSL_PARAM params[]);
int EVP_PKEY_verify_recover(EVP_PKEY_CTX *ctx,
                            unsigned char *rout, size_t *routlen,
                            const unsigned char *sig, size_t siglen);
```

DESCRIPTION

EVP_PKEY_verify_recover_init() initializes a public key algorithm context *ctx* for signing using the algorithm given when the context was created using **EVP_PKEY_CTX_new**(3) or variants thereof. The algorithm is used to fetch a **EVP_SIGNATURE** method implicitly, see "Implicit fetch" in **provider**(7) for more information about implicit fetches.

EVP_PKEY_verify_recover_init_ex() is the same as **EVP_PKEY_verify_recover_init()** but additionally sets the passed parameters *params* on the context before returning.

The **EVP_PKEY_verify_recover()** function recovers signed data using *ctx*. The signature is specified using the *sig* and *siglen* parameters. If *rout* is NULL then the maximum size of the output buffer is written to the *routlen* parameter. If *rout* is not NULL then before the call the *routlen* parameter should contain the length of the *rout* buffer, if the call is successful recovered data is written to *rout* and the amount of data written to *routlen*.

NOTES

Normally an application is only interested in whether a signature verification operation is successful in those cases the **EVP_verify()** function should be used.

Sometimes however it is useful to obtain the data originally signed using a signing operation. Only certain public key algorithms can recover a signature in this way (for example RSA in PKCS padding mode).

After the call to **EVP_PKEY_verify_recover_init()** algorithm specific control operations can be performed to set any appropriate parameters for the operation.

The function **EVP_PKEY_verify_recover()** can be called more than once on the same context if several operations are performed using the same parameters.

RETURN VALUES

EVP_PKEY_verify_recover_init() and **EVP_PKEY_verify_recover()** 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

Recover digest originally signed using PKCS#1 and SHA256 digest:

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

EVP_PKEY_CTX *ctx;
unsigned char *rout, *sig;
size_t routlen, siglen;
EVP_PKEY *verify_key;

/*
 * NB: assumes verify_key, sig and siglen are already set up
 * and that verify_key is an RSA public key
 */
ctx = EVP_PKEY_CTX_new(verify_key, NULL /* no engine */);
if (!ctx)
    /* Error occurred */
if (EVP_PKEY_verify_recover_init(ctx) <= 0)
    /* Error */
if (EVP_PKEY_CTX_set_rsa_padding(ctx, RSA_PKCS1_PADDING) <= 0)
    /* Error */
if (EVP_PKEY_CTX_set_signature_md(ctx, EVP_sha256()) <= 0)
    /* Error */

/* Determine buffer length */
if (EVP_PKEY_verify_recover(ctx, NULL, &routlen, sig, siglen) <= 0)
    /* Error */

rout = OPENSSL_malloc(routlen);

if (!rout)
```

```
    /* malloc failure */

 if (EVP_PKEY_verify_recover(ctx, rout, &routlen, sig, siglen) <= 0)
    /* Error */

 /* Recovered data is routlen bytes written to buffer rout */
```

SEE ALSO

EVP_PKEY_CTX_new(3), **EVP_PKEY_encrypt**(3), **EVP_PKEY_decrypt**(3), **EVP_PKEY_sign**(3), **EVP_PKEY_verify**(3), **EVP_PKEY_derive**(3)

HISTORY

The **EVP_PKEY_verify_recover_init()** and **EVP_PKEY_verify_recover()** functions were added in OpenSSL 1.0.0.

The **EVP_PKEY_verify_recover_init_ex()** function was added in OpenSSL 3.0.

COPYRIGHT