

**NAME**

EVP\_PKEY\_sign\_init, EVP\_PKEY\_sign\_init\_ex, EVP\_PKEY\_sign - sign using a public key algorithm

**SYNOPSIS**

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

```
int EVP_PKEY_sign_init(EVP_PKEY_CTX *ctx);
int EVP_PKEY_sign_init_ex(EVP_PKEY_CTX *ctx, const OSSL_PARAM params[]);
int EVP_PKEY_sign(EVP_PKEY_CTX *ctx,
                 unsigned char *sig, size_t *siglen,
                 const unsigned char *tbs, size_t tbslen);
```

**DESCRIPTION**

**EVP\_PKEY\_sign\_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\_sign\_init\_ex()** is the same as **EVP\_PKEY\_sign\_init()** but additionally sets the passed parameters *params* on the context before returning.

The **EVP\_PKEY\_sign()** function performs a public key signing operation using *ctx*. The data to be signed is specified using the *tbs* and *tbslen* parameters. If *sig* is NULL then the maximum size of the output buffer is written to the *siglen* parameter. If *sig* is not NULL then before the call the *siglen* parameter should contain the length of the *sig* buffer, if the call is successful the signature is written to *sig* and the amount of data written to *siglen*.

**NOTES**

**EVP\_PKEY\_sign()** does not hash the data to be signed, and therefore is normally used to sign digests. For signing arbitrary messages, see the **EVP\_DigestSignInit(3)** and **EVP\_SignInit(3)** signing interfaces instead.

After the call to **EVP\_PKEY\_sign\_init()** algorithm specific control operations can be performed to set any appropriate parameters for the operation (see **EVP\_PKEY\_CTX\_ctrl(3)**).

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

**RETURN VALUES**

**EVP\_PKEY\_sign\_init()** and **EVP\_PKEY\_sign()** 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

Sign data using RSA with PKCS#1 padding and SHA256 digest:

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

EVP_PKEY_CTX *ctx;
/* md is a SHA-256 digest in this example. */
unsigned char *md, *sig;
size_t mdlen = 32, siglen;
EVP_PKEY *signing_key;

/*
 * NB: assumes signing_key and md are set up before the next
 * step. signing_key must be an RSA private key and md must
 * point to the SHA-256 digest to be signed.
 */
ctx = EVP_PKEY_CTX_new(signing_key, NULL /* no engine */);
if (!ctx)
    /* Error occurred */
if (EVP_PKEY_sign_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_sign(ctx, NULL, &siglen, md, mdlen) <= 0)
    /* Error */

sig = OPENSSL_malloc(siglen);

if (!sig)
    /* malloc failure */
```

```
if (EVP_PKEY_sign(ctx, sig, &siglen, md, mdlen) <= 0)
    /* Error */

/* Signature is siglen bytes written to buffer sig */
```

## SEE ALSO

**EVP\_PKEY\_CTX\_new(3), EVP\_PKEY\_CTX\_ctrl(3), EVP\_PKEY\_encrypt(3),  
EVP\_PKEY\_decrypt(3), EVP\_PKEY\_verify(3), EVP\_PKEY\_verify\_recover(3),  
EVP\_PKEY\_derive(3)**

## HISTORY

The **EVP\_PKEY\_sign\_init()** and **EVP\_PKEY\_sign()** functions were added in OpenSSL 1.0.0.

The **EVP\_PKEY\_sign\_init\_ex()** function was added in OpenSSL 3.0.

## COPYRIGHT

Copyright 2006-2021 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>>.