



*Full credit is given to the above companies including the OS that this PDF file was generated!*

***Rocky Enterprise Linux 9.2 Manual Pages on command 'provider-asym\_cipher.7oss1'***

***\$ man provider-asym\_cipher.7oss1***

PROVIDER-ASYM\_CIPHER(7oss1)    OpenSSL    PROVIDER-ASYM\_CIPHER(7oss1)

NAME

provider-asym\_cipher - The asym\_cipher library <-> provider functions

SYNOPSIS

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

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

```
/*
```

```
* None of these are actual functions, but are displayed like this for
```

```
* the function signatures for functions that are offered as function
```

```
* pointers in OSSL_DISPATCH arrays.
```

```
*/
```

```
/* Context management */
```

```
void *OSSL_FUNC_asym_cipher_newctx(void *provctx);
```

```
void OSSL_FUNC_asym_cipher_freectx(void *ctx);
```

```
void *OSSL_FUNC_asym_cipher_dupctx(void *ctx);
```

```
/* Encryption */
```

```
int OSSL_FUNC_asym_cipher_encrypt_init(void *ctx, void *provkey,
```

```
const OSSL_PARAM params[]);
```

```

int OSSL_FUNC_asym_cipher_encrypt(void *ctx, unsigned char *out, size_t *outlen,
    size_t outsize, const unsigned char *in,
    size_t inlen);

/* Decryption */

int OSSL_FUNC_asym_cipher_decrypt_init(void *ctx, void *provkey,
    const OSSL_PARAM params[]);

int OSSL_FUNC_asym_cipher_decrypt(void *ctx, unsigned char *out, size_t *outlen,
    size_t outsize, const unsigned char *in,
    size_t inlen);

/* Asymmetric Cipher parameters */

int OSSL_FUNC_asym_cipher_get_ctx_params(void *ctx, OSSL_PARAM params[]);
const OSSL_PARAM *OSSL_FUNC_asym_cipher_gettable_ctx_params(void *provctx);
int OSSL_FUNC_asym_cipher_set_ctx_params(void *ctx, const OSSL_PARAM params[]);
const OSSL_PARAM *OSSL_FUNC_asym_cipher_settable_ctx_params(void *provctx);

```

## DESCRIPTION

This documentation is primarily aimed at provider authors. See [provider\(7\)](#) for further information.

The asymmetric cipher (`OSSL_OP_ASYM_CIPHER`) operation enables providers to implement asymmetric cipher algorithms and make them available to applications via the API functions `EVP_PKEY_encrypt(3)`, `EVP_PKEY_decrypt(3)` and other related functions).

All "functions" mentioned here are passed as function pointers between `libcrypto` and the provider in `OSSL_DISPATCH` arrays via `OSSL_ALGORITHM` arrays that are returned by the provider's `provider_query_operation()` function (see "Provider Functions" in [provider-base\(7\)](#)).

All these "functions" have a corresponding function type definition named `OSSL_FUNC_{name}_fn`, and a helper function to retrieve the function pointer from an `OSSL_DISPATCH` element named `OSSL_FUNC_{name}`.

For example, the "function" `OSSL_FUNC_asym_cipher_newctx()` has these:

```

typedef void *(OSSL_FUNC_asym_cipher_newctx_fn)(void *provctx);
static ossl_inline OSSL_FUNC_asym_cipher_newctx_fn
    OSSL_FUNC_asym_cipher_newctx(const OSSL_DISPATCH *opf);

```

`OSSL_DISPATCH` arrays are indexed by numbers that are provided as macros

in openssl-core\_dispatch.h(7), as follows:

OSSL_FUNC_asym_cipher_newctx	OSSL_FUNC_ASYM_CIPHER_NEWCTX
OSSL_FUNC_asym_cipher_freectx	OSSL_FUNC_ASYM_CIPHER_FREECTX
OSSL_FUNC_asym_cipher_dupctx	OSSL_FUNC_ASYM_CIPHER_DUPCTX
OSSL_FUNC_asym_cipher_encrypt_init	OSSL_FUNC_ASYM_CIPHER_ENCRYPT_INIT
OSSL_FUNC_asym_cipher_encrypt	OSSL_FUNC_ASYM_CIPHER_ENCRYPT
OSSL_FUNC_asym_cipher_decrypt_init	OSSL_FUNC_ASYM_CIPHER_DECRYPT_INIT
OSSL_FUNC_asym_cipher_decrypt	OSSL_FUNC_ASYM_CIPHER_DECRYPT
OSSL_FUNC_asym_cipher_get_ctx_params	OSSL_FUNC_ASYM_CIPHER_GET_CTX_PARAMS
OSSL_FUNC_asym_cipher_gettable_ctx_params	OSSL_FUNC_ASYM_CIPHER_GETTABLE_CTX_PARAMS
OSSL_FUNC_asym_cipher_set_ctx_params	OSSL_FUNC_ASYM_CIPHER_SET_CTX_PARAMS
OSSL_FUNC_asym_cipher_settable_ctx_params	OSSL_FUNC_ASYM_CIPHER_SETTABLE_CTX_PARAMS

An asymmetric cipher algorithm implementation may not implement all of these functions. In order to be a consistent set of functions a provider must implement OSSL\_FUNC\_asym\_cipher\_newctx and OSSL\_FUNC\_asym\_cipher\_freectx. It must also implement both of OSSL\_FUNC\_asym\_cipher\_encrypt\_init and OSSL\_FUNC\_asym\_cipher\_encrypt, or both of OSSL\_FUNC\_asym\_cipher\_decrypt\_init and OSSL\_FUNC\_asym\_cipher\_decrypt. OSSL\_FUNC\_asym\_cipher\_get\_ctx\_params is optional but if it is present then so must OSSL\_FUNC\_asym\_cipher\_gettable\_ctx\_params. Similarly, OSSL\_FUNC\_asym\_cipher\_set\_ctx\_params is optional but if it is present then so must OSSL\_FUNC\_asym\_cipher\_settable\_ctx\_params. An asymmetric cipher algorithm must also implement some mechanism for generating, loading or importing keys via the key management (OSSL\_OP\_KEYMGMT) operation. See provider-keymgmt(7) for further details.

## Context Management Functions

OSSL\_FUNC\_asym\_cipher\_newctx() should create and return a pointer to a provider side structure for holding context information during an asymmetric cipher operation. A pointer to this context will be passed back in a number of the other asymmetric cipher operation function calls. The parameter provctx is the provider context generated during

provider initialisation (see provider(7)).

OSSL\_FUNC\_asym\_cipher\_freectx() is passed a pointer to the provider side asymmetric cipher context in the ctx parameter. This function should free any resources associated with that context.

OSSL\_FUNC\_asym\_cipher\_dupctx() should duplicate the provider side asymmetric cipher context in the ctx parameter and return the duplicate copy.

#### Encryption Functions

OSSL\_FUNC\_asym\_cipher\_encrypt\_init() initialises a context for an asymmetric encryption given a provider side asymmetric cipher context in the ctx parameter, and a pointer to a provider key object in the provkey parameter. The params, if not NULL, should be set on the context in a manner similar to using

OSSL\_FUNC\_asym\_cipher\_set\_ctx\_params(). The key object should have been previously generated, loaded or imported into the provider using the key management (OSSL\_OP\_KEYMGMT) operation (see provider-keymgmt(7)). OSSL\_FUNC\_asym\_cipher\_encrypt() performs the actual encryption itself. A previously initialised asymmetric cipher context is passed in the ctx parameter. The data to be encrypted is pointed to by the in parameter which is inlen bytes long. Unless out is NULL, the encrypted data should be written to the location pointed to by the out parameter and it should not exceed outsize bytes in length. The length of the encrypted data should be written to \*outlen. If out is NULL then the maximum length of the encrypted data should be written to \*outlen.

#### Decryption Functions

OSSL\_FUNC\_asym\_cipher\_decrypt\_init() initialises a context for an asymmetric decryption given a provider side asymmetric cipher context in the ctx parameter, and a pointer to a provider key object in the provkey parameter. The params, if not NULL, should be set on the context in a manner similar to using

OSSL\_FUNC\_asym\_cipher\_set\_ctx\_params(). The key object should have been previously generated, loaded or imported into the provider using

the key management (OSSL\_OP\_KEYMGMT) operation (see provider-keymgmt(7)).

OSSL\_FUNC\_asym\_cipher\_decrypt() performs the actual decryption itself.

A previously initialised asymmetric cipher context is passed in the ctx parameter. The data to be decrypted is pointed to by the in parameter which is inlen bytes long. Unless out is NULL, the decrypted data should be written to the location pointed to by the out parameter and it should not exceed outsize bytes in length. The length of the decrypted data should be written to \*outlen. If out is NULL then the maximum length of the decrypted data should be written to \*outlen.

### Asymmetric Cipher Parameters

See OSSL\_PARAM(3) for further details on the parameters structure used by the OSSL\_FUNC\_asym\_cipher\_get\_ctx\_params() and OSSL\_FUNC\_asym\_cipher\_set\_ctx\_params() functions.

OSSL\_FUNC\_asym\_cipher\_get\_ctx\_params() gets asymmetric cipher parameters associated with the given provider side asymmetric cipher context ctx and stores them in params. Passing NULL for params should return true.

OSSL\_FUNC\_asym\_cipher\_set\_ctx\_params() sets the asymmetric cipher parameters associated with the given provider side asymmetric cipher context ctx to params. Any parameter settings are additional to any that were previously set. Passing NULL for params should return true.

Parameters currently recognised by built-in asymmetric cipher algorithms are as follows. Not all parameters are relevant to, or are understood by all asymmetric cipher algorithms:

"pad-mode" (OSSL\_ASYM\_CIPHER\_PARAM\_PAD\_MODE) <integer>

The type of padding to be used. The interpretation of this value will depend on the algorithm in use. The default provider understands these RSA padding modes: 1 (RSA\_PKCS1\_PADDING), 3 (RSA\_NO\_PADDING), 4 (RSA\_PKCS1\_OAEP\_PADDING), 5 (RSA\_X931\_PADDING), 6 (RSA\_PKCS1\_PSS\_PADDING) and 7 (RSA\_PKCS1\_WITH\_TLS\_PADDING). See EVP\_PKEY\_CTX\_set\_rsa\_padding(3) for further details.

"digest" (OSSL\_ASYM\_CIPHER\_PARAM\_OAEP\_DIGEST) <UTF8 string>

Gets or sets the name of the OAEP digest algorithm used when OAEP padding is in use.

"digest" (OSSL\_ASYM\_CIPHER\_PARAM\_DIGEST) <UTF8 string>

Gets or sets the name of the digest algorithm used by the algorithm (where applicable).

"digest-props" (OSSL\_ASYM\_CIPHER\_PARAM\_OAEP\_DIGEST\_PROPS) <UTF8 string>

Gets or sets the properties to use when fetching the OAEP digest algorithm.

"digest-props" (OSSL\_ASYM\_CIPHER\_PARAM\_DIGEST\_PROPS) <UTF8 string>

Gets or sets the properties to use when fetching the cipher digest algorithm.

"mgf1-digest" (OSSL\_ASYM\_CIPHER\_PARAM\_MGF1\_DIGEST) <UTF8 string>

Gets or sets the name of the MGF1 digest algorithm used when OAEP or PSS padding is in use.

"mgf1-digest-props" (OSSL\_ASYM\_CIPHER\_PARAM\_MGF1\_DIGEST\_PROPS) <UTF8 string>

Gets or sets the properties to use when fetching the MGF1 digest algorithm.

"oaep-label" (OSSL\_ASYM\_CIPHER\_PARAM\_OAEP\_LABEL) <octet string>

Gets or sets the OAEP label used when OAEP padding is in use.

"tls-client-version" (OSSL\_ASYM\_CIPHER\_PARAM\_TLS\_CLIENT\_VERSION)

<unsigned integer>

The TLS protocol version first requested by the client. See `RSA_PKCS1_WITH_TLS_PADDING` on the page `EVP_PKEY_CTX_set_rsa_padding(3)`.

"tls-negotiated-version" (OSSL\_ASYM\_CIPHER\_PARAM\_TLS\_CLIENT\_VERSION)

<unsigned integer>

The negotiated TLS protocol version. See `RSA_PKCS1_WITH_TLS_PADDING` on the page `EVP_PKEY_CTX_set_rsa_padding(3)`.

`OSSL_FUNC_asym_cipher_gettable_ctx_params()` and

`OSSL_FUNC_asym_cipher_settable_ctx_params()` get a constant `OSSL_PARAM` array that describes the gettable and settable parameters, i.e.

parameters that can be used with `OSSL_FUNC_asym_cipher_get_ctx_params()`

and `OSSL_FUNC_asym_cipher_set_ctx_params()` respectively. See `OSSL_PARAM(3)` for the use of `OSSL_PARAM` as parameter descriptor.

## RETURN VALUES

`OSSL_FUNC_asym_cipher_newctx()` and `OSSL_FUNC_asym_cipher_dupctx()` should return the newly created provider side asymmetric cipher context, or `NULL` on failure.

All other functions should return 1 for success or 0 on error.

## SEE ALSO

`provider(7)`

## HISTORY

The provider `ASYM_CIPHER` interface was introduced in OpenSSL 3.0.

## COPYRIGHT

Copyright 2019-2021 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>.

3.0.7                    2023-07-13    PROVIDER-ASYM\_CIPHER(7ossl)