

NAME

enc - symmetric cipher routines

SYNOPSIS

```
openssl enc -ciphername [-in filename] [-out filename] [-pass arg] [-e] [-d] [-a/-base64]
[-A] [-k password] [-kfile filename] [-K key] [-iv IV] [-S salt] [-salt] [-nosalt] [-z] [-md] [-p]
[-P] [-bufsize number] [-nopad] [-debug] [-none] [-engine id]
```

DESCRIPTION

The symmetric cipher commands allow data to be encrypted or decrypted using various block and stream ciphers using keys based on passwords or explicitly provided. Base64 encoding or decoding can also be performed either by itself or in addition to the encryption or decryption.

OPTIONS

-in filename

the input filename, standard input by default.

-out filename

the output filename, standard output by default.

-pass arg

the password source. For more information about the format of **arg** see the **PASS PHRASE ARGUMENTS** section in *openssl(1)*.

-salt

use a salt in the key derivation routines. This is the default.

-nosalt

don't use a salt in the key derivation routines. This option **SHOULD NOT** be used except for test purposes or compatibility with ancient versions of OpenSSL and SSLeay.

-e encrypt the input data: this is the default.

-d decrypt the input data.

-a base64 process the data. This means that if encryption is taking place the data is base64 encoded after encryption. If decryption is set then the input data is base64 decoded before being decrypted.

-base64

same as **-a**

-A if the **-a** option is set then base64 process the data on one line.

-k password

the password to derive the key from. This is for compatibility with previous versions of OpenSSL. Superseded by the **-pass** argument.

-kfile filename

read the password to derive the key from the first line of **filename**. This is for compatibility with previous versions of OpenSSL. Superseded by the **-pass** argument.

-nosalt

do not use a salt

-salt

use salt (randomly generated or provide with **-S** option) when encrypting (this is the default).

-S salt

the actual salt to use: this must be represented as a string of hex digits.

-K key

the actual key to use: this must be represented as a string comprised only of hex digits. If only the key is specified, the IV must additionally specified using the **-iv** option. When both

a key and a password are specified, the key given with the **-K** option will be used and the IV generated from the password will be taken. It probably does not make much sense to specify both key and password.

-iv IV

the actual IV to use: this must be represented as a string comprised only of hex digits. When only the key is specified using the **-K** option, the IV must explicitly be defined. When a password is being specified using one of the other options, the IV is generated from this password.

-p print out the key and IV used.

-P print out the key and IV used then immediately exit: don't do any encryption or decryption.

-bufsize number

set the buffer size for I/O

-nopad

disable standard block padding

-debug

debug the BIOs used for I/O.

-z Compress or decompress clear text using zlib before encryption or after decryption. This option exists only if OpenSSL with compiled with zlib or zlib-dynamic option.

-none

Use NULL cipher (no encryption or decryption of input).

NOTES

The program can be called either as **openssl ciphername** or **openssl enc -ciphername**. But the first form doesn't work with engine-provided ciphers, because this form is processed before the configuration file is read and any ENGINES loaded.

Engines which provide entirely new encryption algorithms (such as ccbgost engine which provides gost89 algorithm) should be configured in the configuration file. Engines, specified in the command line using **-engine** options can only be used for hardware-assisted implementations of ciphers, which are supported by OpenSSL core or other engine, specified in the configuration file.

When enc command lists supported ciphers, ciphers provided by engines, specified in the configuration files are listed too.

A password will be prompted for to derive the key and IV if necessary.

The **-salt** option should **ALWAYS** be used if the key is being derived from a password unless you want compatibility with previous versions of OpenSSL and SSLeay.

Without the **-salt** option it is possible to perform efficient dictionary attacks on the password and to attack stream cipher encrypted data. The reason for this is that without the salt the same password always generates the same encryption key. When the salt is being used the first eight bytes of the encrypted data are reserved for the salt: it is generated at random when encrypting a file and read from the encrypted file when it is decrypted.

Some of the ciphers do not have large keys and others have security implications if not used correctly. A beginner is advised to just use a strong block cipher in CBC mode such as bf or des3.

All the block ciphers normally use PKCS#5 padding also known as standard block padding: this allows a rudimentary integrity or password check to be performed. However since the chance of random data passing the test is better than 1 in 256 it isn't a very good test.

If padding is disabled then the input data must be a multiple of the cipher block length.

All RC2 ciphers have the same key and effective key length.

Blowfish and RC5 algorithms use a 128 bit key.

SUPPORTED CIPHERS

Note that some of these ciphers can be disabled at compile time and some are available only if an appropriate engine is configured in the configuration file. The output of the **enc** command run with unsupported options (for example **openssl enc -help**) includes a list of ciphers, supported by your version of OpenSSL, including ones provided by configured engines.

The **enc** program does not support authenticated encryption modes like CCM and GCM. The utility does not store or retrieve the authentication tag.

base64 Base 64

bf-cbc Blowfish in CBC mode
 bf Alias for bf-cbc
 bf-cfb Blowfish in CFB mode
 bf-ecb Blowfish in ECB mode
 bf-ofb Blowfish in OFB mode

cast-cbc CAST in CBC mode
 cast Alias for cast-cbc
 cast5-cbc CAST5 in CBC mode
 cast5-cfb CAST5 in CFB mode
 cast5-ecb CAST5 in ECB mode
 cast5-ofb CAST5 in OFB mode

des-cbc DES in CBC mode
 des Alias for des-cbc
 des-cfb DES in CBC mode
 des-ofb DES in OFB mode
 des-ecb DES in ECB mode

des-ede-cbc Two key triple DES EDE in CBC mode
 des-ede Two key triple DES EDE in ECB mode
 des-ede-cfb Two key triple DES EDE in CFB mode
 des-ede-ofb Two key triple DES EDE in OFB mode

des-ede3-cbc Three key triple DES EDE in CBC mode
 des-ede3 Three key triple DES EDE in ECB mode
 des3 Alias for des-ede3-cbc
 des-ede3-cfb Three key triple DES EDE CFB mode
 des-ede3-ofb Three key triple DES EDE in OFB mode

desx DESX algorithm.

gost89 GOST 28147-89 in CFB mode (provided by ccgost engine)
 gost89-cnt `GOST 28147-89 in CNT mode (provided by ccgost engine)

idea-cbc IDEA algorithm in CBC mode
 idea same as idea-cbc
 idea-cfb IDEA in CFB mode
 idea-ecb IDEA in ECB mode
 idea-ofb IDEA in OFB mode

rc2-cbc 128 bit RC2 in CBC mode
 rc2 Alias for rc2-cbc
 rc2-cfb 128 bit RC2 in CFB mode
 rc2-ecb 128 bit RC2 in ECB mode

```

rc2-ofb 128 bit RC2 in OFB mode
rc2-64-cbc 64 bit RC2 in CBC mode
rc2-40-cbc 40 bit RC2 in CBC mode

rc4 128 bit RC4
rc4-64 64 bit RC4
rc4-40 40 bit RC4

rc5-cbc RC5 cipher in CBC mode
rc5 Alias for rc5-cbc
rc5-cfb RC5 cipher in CFB mode
rc5-ecb RC5 cipher in ECB mode
rc5-ofb RC5 cipher in OFB mode

aes-[128|192|256]-cbc 128/192/256 bit AES in CBC mode
aes-[128|192|256] Alias for aes-[128|192|256]-cbc
aes-[128|192|256]-cfb 128/192/256 bit AES in 128 bit CFB mode
aes-[128|192|256]-cfb1 128/192/256 bit AES in 1 bit CFB mode
aes-[128|192|256]-cfb8 128/192/256 bit AES in 8 bit CFB mode
aes-[128|192|256]-ecb 128/192/256 bit AES in ECB mode
aes-[128|192|256]-ofb 128/192/256 bit AES in OFB mode

```

EXAMPLES

Just base64 encode a binary file:

```
openssl base64 -in file.bin -out file.b64
```

Decode the same file

```
openssl base64 -d -in file.b64 -out file.bin
```

Encrypt a file using triple DES in CBC mode using a prompted password:

```
openssl des3 -salt -in file.txt -out file.des3
```

Decrypt a file using a supplied password:

```
openssl des3 -d -salt -in file.des3 -out file.txt -k mypassword
```

Encrypt a file then base64 encode it (so it can be sent via mail for example) using Blowfish in CBC mode:

```
openssl bf -a -salt -in file.txt -out file.bf
```

Base64 decode a file then decrypt it:

```
openssl bf -d -salt -a -in file.bf -out file.txt
```

Decrypt some data using a supplied 40 bit RC4 key:

```
openssl rc4-40 -in file.rc4 -out file.txt -K 0102030405
```

BUGS

The **-A** option when used with large files doesn't work properly.

There should be an option to allow an iteration count to be included.

The **enc** program only supports a fixed number of algorithms with certain parameters. So if, for example, you want to use RC2 with a 76 bit key or RC4 with an 84 bit key you can't use this program.