# Encrypt And Decrypt Files With A Password Using OpenSSL

OpenSSL (/en/Categories/Cryptography/OpenSSL) ;

**OpenSSL** is a powerful cryptography toolkit that can be used for **encryption of files and messages**.

If you want to use the same password for both **encryption of plaintext** and **decryption of ciphertext**, then you have to use a method that is known as **symmetric-key** algorithm.

From this article you'll learn how to **encrypt and decrypt files and messages** with a password from the Linux command line, using OpenSSL.

## HowTo : Encrypt a File

```
$ openssl enc -aes-256-cbc -salt -in file.txt -out file.txt.enc
```

| Options | Description |
|---|---|
| openssl | OpenSSL command line tool. |
| enc | Encoding with Ciphers. |
| -aes-256-cbc | The encryption cipher to be used. |
| -salt | Adds strength to the encryption. |
| -in | Specifies the input file. |
| -out | Specifies the output file. |

256bit AES is what the United States government uses to encrypt information at the Top Secret level.

The **-salt** option should **ALWAYS** be used if the key is being derived from a password.

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.

## HowTo : Decrypt a File

```
$ openssl enc -aes-256-cbc -d -in file.txt.enc -out file.txt
```

| Options | Description |
|---|---|
| -d | Decrypts data. |
| -in | Specifies the data to decrypt. |
| -out | Specifies the file to put the decrypted data in. |

## Base64 Encode and Decode

Base64 encoding is a standard method for converting 8-bit binary information into a limited subset of ASCII characters.

It is needed for safe transport through e-mail systems, and other systems that are not 8-bit safe.

By default the encrypted file is in a binary format. If you are going to send it by email, IRC, etc. you have to save encrypted file in Base64-encode.

To encrypt file in Base64-encode, you should add **-a** option :

```
$ openssl enc -aes-256-cbc -salt -a -in file.txt -out file.txt.enc
```

| Option | Description |
|---|---|
| -a | Tells OpenSSL that the encrypted data is in Base64-ensode. |

Option **-a** should also be added while decryption :

```
$ openssl enc -aes-256-cbc -d -a -in file.txt.enc -out file.txt
```

## Non interactive Encrypt / Decrypt

Since the password is visible, this form should only be used where **security is not important**.

By default a user is prompted to enter the password.

If you are creating a BASH script, you may want to set the password in non interactive way, using **-k** option.

Public key cryptography was invented just for such cases.

Encrypt a file using a supplied password :

```
$ openssl enc -aes-256-cbc -salt -in file.txt -out file.txt.enc -k PASS
```

Decrypt a file using a supplied password :

```
$ openssl enc -aes-256-cbc -d -in file.txt.enc -out file.txt -k PASS
```

password (/en/Tags/password) decryption (/en/Tags/decryption) encryption (/en/Tags/encryption) OpenSSL (/en/Tags/OpenSSL)

## RELATED ARTICLES

- HowTo : Create a Password Protected ZIP File in Linux (/en/HowTo-Create-a-Password-Protected-ZIP-File-in-Linux)
- /etc/shadow - HowTo: Generate Password Hash in Linux (/en/etc-shadow-HowTo-Generate-Password-Hash-in-Linux)
- Creating Certificate Signing Request -- CSR Generation (/en/Creating-Certificate-Signing-Request-CSR-Generation)
- HowTo : Decode SSL Certificate (/en/HowTo-Decode-SSL-Certificate)
- HowTo : Decode CSR (/en/HowTo-Decode-CSR)
- Moving SSL Certificate from IIS to Apache (/en/Moving-SSL-Certificate-from-IIS-to-Apache)
- HowTo : Find a Key Length from the Linux Shell using OpenSSL (/en/HowTo-Find-a-Key-Length-from-the-Linux-Shell-using-OpenSSL)

**2 Comments** **www.shellhacks.com** 1 **Login**

**Recommend** **Share** **Sort by Best**

Join the discussion…

**Nneko Branche** • 2 years ago

We also did a similar encryption tutorial. It does into more details and provides some background. You can find it here http://www.bigthinkingapplied....

Reply • Share ›

**Renoir Boulanger** • 2 years ago

Hi, that's a very thorough article and I like that you documented more than one useful use-case scenarios with OpenSSL. The software is very good, but documentation is still too cryptic and I felt I could understand better by reading your article.

You brought an interresting use-case here; encoding a file. A question lingered in my mind. Do you know a way to read metadata on an encoded file?

Your example explains very well on how to encode in `aes-256-cbc`. But after digging around it made me wonder what if I forget what cipher was used to encode that file in the first place. (e.g. you encoded the file years ago and lost the note).

What I see is that we should keep a note on the format too I presume.

So far, openssl "standard commands" mentioned in the man pages (e.g. dsa, enc, etc) doesn't have anything that would enable to get some metadata about it.

Update, I found here [0] that we have to remember both the passphrase and the cipher.

[0]: http://www.madboa.com/geek/ope...

Reply • Share ›

**Subscribe** **Add Disqus to your site Add Disqus Add** **Privacy**