How to use OpenSSL to encrypt/decrypt files?

Asked 10 years, 6 months ago Modified 4 months ago Viewed 767k times

308

I want to encrypt and decrypt one file using one password.

How can I use OpenSSL to do that?

encryption openssl

Share Improve this question Follow

edited Oct 23, 2020 at 6:49
Matthias Braun
32.4k 25 144 173

asked Apr 17, 2013 at 9:15
aF.
65.3k 43 136 199

3 You should derive a Key and IV from the password using `PKCS5_PBKDF2_HMAC`. You should use the `EVP_*` functions to encrypt and decrypt. See EVP Symmetric Encryption and Decryption on the OpenSSL wiki. In fact, you should probably be using authenticated encryption because it provides *both* confidentiality and authenticity. See EVP Authenticated Encryption and Decryption on the OpenSSL wiki. – jww May 15, 2015 at 21:28

3 Don't understand from your question why you want OpenSSL. A comment below shows GPG is better - also because of security. stackoverflow.com/a/31552829/952234 I vote down. – Yaroslav Nikitenko Mar 9, 2016 at 13:40

You may like to look at "keepout" so as to save all the encryption options used with the encrypted file... This is especially important now that 'default' options of openssl enc has changed, and will likely change in the future. Also sets a higher and randomised iteration count for the new -pbkdf2 option. antofthy.gitlab.io/software/#keepout – anthony Sep 22, 2020 at 23:48

9 Answers

Sorted by: Highest score (default)

348

Security Warning: AES-256-CBC does not provide authenticated encryption and is vulnerable to padding oracle attacks. You should use something like age instead.

Encrypt:

```
openssl aes-256-cbc -a -salt -pbkdf2 -in secrets.txt -out secrets.txt.enc
```

Decrypt:

```
openssl aes-256-cbc -d -a -pbkdf2 -in secrets.txt.enc -out secrets.txt.new
```

More details on the various flags

Share Improve this answer Follow

edited Dec 6, 2021 at 12:54
Itay Grudev
7,125 4 54 88

answered Apr 17, 2013 at 9:24
Szocske
7,486 2 20 24

37 Answer is likely not optimal (as of this writing) depending on OP's use case. Specifically the parameters "-a" is likely not optimal and the answer does not explain its use. "-a" is typically used when the encrypted output is to be transmitted in ASCII/text form and has the effect of increasing output size compared binary form. The original poster does not specify output format and so I feel that at the very least this should be mentioned. See answer: stackoverflow.com/a/31552829/952234 which also includes a note on why you should use gpg instead of openssl for this task. – moo Mar 26, 2016 at 16:16

12 Do not use the above command since there is no key derivation. Read more here: openssl weak key derivation – Jonas Lejon Apr 29, 2016 at 13:25

1 Should also specify a key, or mention where it comes from. Is it strengthened? – Tuntable May 9, 2016 at 0:41

6 @jonasl according to the latest man page, it states: “The default digest was changed from MD5 to SHA256 in Openssl 1.1.0.” Source: github.com/openssl/openssl/blob/master/doc/man1/enc.pod – Kebman Jul 5, 2018 at 15:43

4 Adding to the comment from @Kebman, you can add `-md sha256` to your encode and decode command if you plan on using this file on another machine. That should cover you against OpenSSL version incompatibilities/differences – dev-rowbot Oct 1, 2018 at 10:03

264

Using GPG (Better Alternative to OpenSSL)

Though you have specifically asked about OpenSSL (see **"Using OpenSSL"** below for OpenSSL solution) you might want to consider using GPG instead for the purpose of encryption based on this article [OpenSSL vs GPG for encrypting off-site backups?](#)

To use GPG to do the same you would use the following commands:

To Encrypt:

```
gpg --output encrypted.data --symmetric --cipher-algo AES256 un_encrypted.data
```

To Decrypt:

```
gpg --output un_encrypted.data --decrypt encrypted.data
```

Note: You will be prompted for a password when encrypting or decrypt. And use `--no-symkey-cache` flag for no cache.

Using OpenSSL (Short Answer)

You likely want to use `gpg` instead of `openssl` as mentioned above but to answer the question using `openssl`:

To Encrypt:

```
openssl enc -aes-256-cbc -in un_encrypted.data -out encrypted.data
```

To Decrypt:

```
openssl enc -d -aes-256-cbc -in encrypted.data -out un_encrypted.data
```

Note: You will be prompted for a password when encrypting or decrypt.

Using OpenSSL (Long Answer)

Your best source of information for `openssl enc` would probably be:
https://www.openssl.org/docs/man1.1.1/man1/enc.html

Command line: `openssl enc` takes the following form:

```
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]
```

Explanation of most useful parameters with regards to your question:

```
-e
    Encrypt the input data: this is the default.

-d
    Decrypt the input data.

-k <password>
    Only use this if you want to pass the password as an argument.
    Usually you can leave this out and you will be prompted for a
    password. The password is used to derive the actual key which
    is used to encrypt your data. Using this parameter is typically
    not considered secure because your password appears in
    plain-text on the command line and will likely be recorded in
    bash history.

-kfile <filename>
    Read the password from the first line of <filename> instead of
    from the command line as above.

-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.
    You likely DON'T need to use this. This will likely increase the
    file size for non-text data. Only use this if you need to send
    data in the form of text format via email etc.

-salt
    To use a salt (randomly generated) when encrypting. You always
    want to use a salt while encrypting. This parameter is actually
    redundant because a salt is used whether you use this or not
    which is why it was not used in the "Short Answer" above!

-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 be specified using the -iv option. When
```

Share Improve this answer Follow — edited Jun 15 at 21:30 — answered Jul 22, 2015 at 2:24 moo 2,978 1 12 10

16 Great comment about preferring GPG over OpenSSL. I find it incredible that OpenSSL uses such a weak password derived hash for the key! – Mark Oct 26, 2016 at 14:23

4 Be sure to use the "-md md5" option for compatibility with files that were encrypted on older openssl without the -md option specified, otherwise you will find that files won't decrypt on newer systems: github.com/libressl-portable/portable/issues/378 – Sam Liddicott Jul 18, 2018 at 15:51

11 "You will be prompted for a password when encrypting or decrypt." `gpg` is letting me decrypt a file without being prompted for a password. It looks like the password is stored for some period of time, which I don't want. – user76284 Jul 30, 2019 at 2:45

3 @user76284 This might address your issue: unix.stackexchange.com/a/395876/79875 . I think you have `gpg-agent` running – moo Jul 31, 2019 at 16:05

4 @moo It also seems that the option `--no-symkey-cache` disables caching when using gpg with

50

Encrypt:

```
openssl enc -in infile.txt -out encrypted.dat -e -aes256 -k symmetrickey
```

Decrypt:

```
openssl enc -in encrypted.dat -out outfile.txt -d -aes256 -k symmetrickey
```

For details, see the `openssl(1)` docs.

Share Improve this answer Follow

edited Aug 26, 2021 at 16:14
Andrii Abramov
10.1k 10 74 97

answered Apr 17, 2013 at 9:29
Ken Cheung
1,778 14 13

17 To use a plaintext password, replace `-k symmetrickey` with `-pass stdin` or `-pass 'pass:PASSWORD'` – Zenexer Feb 14, 2015 at 5:35

6 Do not use the above command since there is no key derivation. Read more here: openssl weak key derivation – Jonas Lejon Apr 29, 2016 at 13:26

8 Related to @jonasl's comment, note that `-k symmetrickey` is misleading. The `-k` option is used for specifying a password, from which OpenSSL derives the symmetric key. If you want to specify the symmetric key, you must use the `-K` option. – user1071847 Aug 22, 2017 at 21:17

28

DO NOT USE OPENSSL DEFAULT KEY DERIVATION.

Currently the accepted answer makes use of it and it's no longer recommended and secure.

It is very feasible for an attacker to simply brute force the key.

https://www.ietf.org/rfc/rfc2898.txt

> PBKDF1 applies a hash function, which shall be MD2 [6], MD5 [19] or SHA-1 [18], to derive keys. The length of the derived key is bounded by the length of the hash function output, which is 16 octets for MD2 and MD5 and 20 octets for SHA-1. PBKDF1 is compatible with the key derivation process in PKCS #5 v1.5. PBKDF1 is recommended only for compatibility with existing applications since the keys it produces may not be large enough for some applications.
>
> PBKDF2 applies a pseudorandom function (see Appendix B.1 for an example) to derive keys. The length of the derived key is essentially unbounded. (However, the maximum effective search space for the derived key may be limited by the structure of the underlying pseudorandom function. See Appendix B.1 for further discussion.) PBKDF2 is recommended for new applications.

Do this:

```
openssl enc -aes-256-cbc -pbkdf2 -iter 20000 -in hello -out hello.enc -k meow
```

```
openssl enc -d -aes-256-cbc -pbkdf2 -iter 20000 -in hello.enc -out hello.out
```

Note: Iterations in decryption have to be the same as iterations in encryption.

Iterations have to be a minimum of 10000. Here is a good answer on the number of iterations: https://security.stackexchange.com/a/3993

Also... we've got enough people here recommending GPG. Read the damn question.

Share Improve this answer Follow

answered May 3, 2019 at 18:46
Arnold Balliu
1,099 10 21

NOTE: PBKDF2 is now part of the openssl enc (finally). However the iteration count is extremenly low, and needs to be set to a much higher level. If that count is randomised, then you also get a extra level of 'saltiness' to your encryption. – anthony Sep 22, 2020 at 23:45

The linked article is great. It also hints that instead of picking high count, which increases computational resources linearly, one can simply use strong password with high entropy, causing computational complexity to grow exponentially. Every extra 10 bits of entropy is equivalent to multiplying iter count by 1000. E.g. if you have >28 random characters (from a set of 62) as a password you don't need to worry about the iteration count altogether. – ɔɥǝɹ̣ıo Feb 3, 2022 at 18:57

It would be really beneficial if below those two statements, that you defined all the switches and commands used. – Andrew S May 30 at 4:31

What is the -k? Its not in the man pages. – Andrew S May 30 at 4:45

@AndrewS - fist line: use `aes-256-cbc` as the "Cypher command", `-pbkdf2` is the "Password-Based Key Derivation Function, 2nd gen.", `-iter` is the number of iterations the key derivation function runs, `-in` input file, `-out` result of encryption/decryption, `-k` literally the password used. The second command will prompt for the password, which is `meow` in this case. For your sake, use a vastly longer passphrase than `meow` ;). Note I usually use `-pass file:keyfile` and `keyfile` is usually a 64+ character file. You'll need that to decrypt, so don't lose it! – Jon V Sep 14 at 19:07

15

As mentioned in the other answers, previous versions of openssl used a weak key derivation function to derive an AES encryption key from the password. However, openssl v1.1.1 supports a stronger key derivation function, where the key is derived from the password using `pbkdf2` with a randomly generated salt, and multiple iterations of sha256 hashing (10,000 by default).

To encrypt a file:

```
openssl aes-256-cbc -e -salt -pbkdf2 -iter 10000 -in plaintextfilename -out
encryptedfilename
```

To decrypt a file:

```
openssl aes-256-cbc -d -salt -pbkdf2 -iter 10000 -in encryptedfilename -out
plaintextfilename
```

Note: An equivalent/compatible implementation in javascript (using the web crypto api) can be found at https://github.com/meixler/web-browser-based-file-encryption-decryption.

Share Improve this answer Follow

edited Aug 4, 2022 at 22:25

answered Apr 12, 2020 at 2:43

mti2935
11.6k 3 29 33

Which as these options keep changing, means you need to also keep a record of what options was used when creating each openssl encrypted file. Especially as the iteration count should increase with time! For one solution see as relatively simple wrapper around openssl enc... "keepout" antofthy.gitlab.io/software/#keepout It can expand to include more openssl are time goes on. – anthony May 27, 2020 at 2:47

You need to explain all of those switches, without that its just magic code with no understanding provided. – Andrew S May 30 at 4:41

1 @AndrewS agreed - the openssl man page isn't nearly as nice as most others. You need to *know* that you have to look at `man enc` instead, for the options on using `openssl enc`. – Jon V Sep 14 at 19:16

8

To Encrypt:

```
$ openssl bf < arquivo.txt > arquivo.txt.bf
```

To Decrypt:

```
$ openssl bf -d < arquivo.txt.bf > arquivo.txt
```

bf === Blowfish in CBC mode

Share Improve this answer Follow

answered Jul 31, 2015 at 18:54
Fábio Almeida
189 2 8

This doesn't work in openssl 3, was written. – velcrow Mar 23 at 22:08

8

Update using a random generated public key.

Encypt:

```
openssl enc -aes-256-cbc -a -salt -in {raw data} -out {encrypted data} -pass
file:{random key}
```

Decrypt:

```
openssl enc -d -aes-256-cbc -in {ciphered data} -out {raw data}
```

Share Improve this answer Follow

edited Dec 2, 2020 at 9:48
Ewoks
12.3k 8 58 67

answered Aug 21, 2014 at 23:45
nneko
770 9 11

would be better if that page is still online and it uses https – Ewoks Dec 2, 2020 at 9:48

3

There is an open source program that I find online it uses openssl to encrypt and decrypt files. It does this with a single password. The great thing about this open source script is that it deletes the original unencrypted file by shredding the file. But the dangerous thing about is once the original unencrypted file is gone you have to make sure you remember your password otherwise they be no other way to decrypt your file.

Here the link it is on github

https://github.com/EgbieAnderson1/linux_file_encryptor/blob/master/file_encrypt.py

Share Improve this answer Follow

answered Dec 1, 2014 at 16:06
Michael linkston
39 2

Things have changed when using openssl for file encryption, their are a lot more options, which needs need to be remembers so you can successfully decrypt encrypted files. One solution to this is "keepout" antofthy.gitlab.io/software/#keepout – anthony May 27, 2020 at 2:44

2

Note that the OpenSSL CLI uses a weak non-standard algorithm to convert the passphrase to a key, and installing GPG results in various files added to your home directory and a gpg-agent background process running. If you want maximum portability and control with existing tools, you can use PHP or Python to access the lower-level APIs and directly pass in a full AES Key and IV.

Example PHP invocation via Bash:

```
IV='c2FtcGxlLWFlcy1pdjEyMw=='
KEY='Twsn8eh2w2HbVCF5zKArlY+Mv5ZwVyaGlk5QkeoSlmc='
INPUT=123456789023456

ENCRYPTED=$(php -r "print(openssl_encrypt('$INPUT','aes-256-
ctr',base64_decode('$KEY'),OPENSSL_ZERO_PADDING,base64_decode('$IV')));")
echo '$ENCRYPTED='$ENCRYPTED
DECRYPTED=$(php -r "print(openssl_decrypt('$ENCRYPTED','aes-256-
ctr',base64_decode('$KEY'),OPENSSL_ZERO_PADDING,base64_decode('$IV')));")
echo '$DECRYPTED='$DECRYPTED
```

This outputs:

```
$ENCRYPTED=nzRi252dayEsGXZOTPXW
$DECRYPTED=123456789023456
```

You could also use PHP's `openssl_pbkdf2` function to convert a passphrase to a key securely.

Share Improve this answer Follow

answered Feb 8, 2017 at 2:22
zeroimpl
2,756 22 19

Openssl CLI now implements and warns users that they should use PBKDF2 for password hashing. However its default iteration count is very low, and needs to be much larger. – anthony May 27, 2020 at 2:30