



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 'openssl-pkcs8.1oss1'

\$ man openssl-pkcs8.1oss1

OPENSSL-PKCS8(1oss1) OpenSSL OPENSSL-PKCS8(1oss1)

NAME

openssl-pkcs8 - PKCS#8 format private key conversion command

SYNOPSIS

```
openssl pkcs8 [-help] [-topk8] [-inform DER|PEM] [-outform DER|PEM]
[-in filename] [-passin arg] [-out filename] [-passout arg] [-iter
count] [-noiter] [-nocrypt] [-traditional] [-v2 alg] [-v2prf alg] [-v1
alg] [-scrypt] [-scrypt_N N] [-scrypt_r r] [-scrypt_p p] [-rand files]
[-writerand file] [-engine id] [-provider name] [-provider-path path]
[-propquery propq]
```

DESCRIPTION

This command processes private keys in PKCS#8 format. It can handle both unencrypted PKCS#8 PrivateKeyInfo format and EncryptedPrivateKeyInfo format with a variety of PKCS#5 (v1.5 and v2.0) and PKCS#12 algorithms.

OPTIONS

-help

Print out a usage message.

-topk8

Normally a PKCS#8 private key is expected on input and a private key will be written to the output file. With the -topk8 option the situation is reversed: it reads a private key and writes a PKCS#8 format key.

-inform DER|PEM, -outform DER|PEM

The input and formats; the default is PEM. See `openssl-format-options(1)` for details.

If a key is being converted from PKCS#8 form (i.e. the -topk8 option is not used) then the input file must be in PKCS#8 format.

An encrypted key is expected unless -nocrypt is included.

If -topk8 is not used and PEM mode is set the output file will be an unencrypted private key in PKCS#8 format. If the -traditional option is used then a traditional format private key is written instead.

If -topk8 is not used and DER mode is set the output file will be an unencrypted private key in traditional DER format.

If -topk8 is used then any supported private key can be used for the input file in a format specified by -inform. The output file will be encrypted PKCS#8 format using the specified encryption parameters unless -nocrypt is included.

-traditional

When this option is present and -topk8 is not a traditional format private key is written.

-in filename

This specifies the input filename to read a key from or standard input if this option is not specified. If the key is encrypted a pass phrase will be prompted for.

-passin arg, -passout arg

The password source for the input and output file. For more information about the format of arg see `openssl-passphrase-options(1)`.

-out filename

This specifies the output filename to write a key to or standard output by default. If any encryption options are set then a pass phrase will be prompted for. The output filename should not be the same as the input filename.

-iter count

When creating new PKCS#8 containers, use a given number of iterations on the password in deriving the encryption key for the PKCS#8 output. High values increase the time required to brute-force a PKCS#8 container.

-noiter

When creating new PKCS#8 containers, use 1 as iteration count.

-nocrypt

PKCS#8 keys generated or input are normally PKCS#8 EncryptedPrivateKeyInfo structures using an appropriate password based encryption algorithm. With this option an unencrypted PrivateKeyInfo structure is expected or output. This option does not encrypt private keys at all and should only be used when absolutely necessary. Certain software such as some versions of Java code signing software used unencrypted private keys.

-v2 alg

This option sets the PKCS#5 v2.0 algorithm.

The alg argument is the encryption algorithm to use, valid values include aes128, aes256 and des3. If this option isn't specified then aes256 is used.

-v2prf alg

This option sets the PRF algorithm to use with PKCS#5 v2.0. A typical value value would be hmacWithSHA256. If this option isn't set then the default for the cipher is used or hmacWithSHA256 if there is no default.

Some implementations may not support custom PRF algorithms and may require the hmacWithSHA1 option to work.

-v1 alg

This option indicates a PKCS#5 v1.5 or PKCS#12 algorithm should be

used. Some older implementations may not support PKCS#5 v2.0 and may require this option. If not specified PKCS#5 v2.0 form is used.

`-scrypt`

Uses the scrypt algorithm for private key encryption using default parameters: currently $N=16384$, $r=8$ and $p=1$ and AES in CBC mode with a 256 bit key. These parameters can be modified using the `-scrypt_N`, `-scrypt_r`, `-scrypt_p` and `-v2` options.

`-scrypt_N N`, `-scrypt_r r`, `-scrypt_p p`

Sets the scrypt N , r or p parameters.

`-rand files`, `-writerand file`

See "Random State Options" in `openssl(1)` for details.

`-engine id`

See "Engine Options" in `openssl(1)`. This option is deprecated.

`-provider name`

`-provider-path path`

`-propquery propq`

See "Provider Options" in `openssl(1)`, `provider(7)`, and `property(7)`.

NOTES

By default, when converting a key to PKCS#8 format, PKCS#5 v2.0 using 256 bit AES with HMAC and SHA256 is used.

Some older implementations do not support PKCS#5 v2.0 format and require the older PKCS#5 v1.5 form instead, possibly also requiring insecure weak encryption algorithms such as 56 bit DES.

Private keys encrypted using PKCS#5 v2.0 algorithms and high iteration counts are more secure than those encrypted using the traditional SSLeay compatible formats. So if additional security is considered important the keys should be converted.

It is possible to write out DER encoded encrypted private keys in PKCS#8 format because the encryption details are included at an ASN1 level whereas the traditional format includes them at a PEM level.

PKCS#5 V1.5 AND PKCS#12 ALGORITHMS

Various algorithms can be used with the `-v1` command line option,

including PKCS#5 v1.5 and PKCS#12. These are described in more detail below.

PBE-MD2-DES PBE-MD5-DES

These algorithms were included in the original PKCS#5 v1.5 specification. They only offer 56 bits of protection since they both use DES.

PBE-SHA1-RC2-64, PBE-MD2-RC2-64, PBE-MD5-RC2-64, PBE-SHA1-DES

These algorithms are not mentioned in the original PKCS#5 v1.5 specification but they use the same key derivation algorithm and are supported by some software. They are mentioned in PKCS#5 v2.0.

They use either 64 bit RC2 or 56 bit DES.

PBE-SHA1-RC4-128, PBE-SHA1-RC4-40, PBE-SHA1-3DES, PBE-SHA1-2DES, PBE-SHA1-RC2-128, PBE-SHA1-RC2-40

These algorithms use the PKCS#12 password based encryption algorithm and allow strong encryption algorithms like triple DES or 128 bit RC2 to be used.

EXAMPLES

Convert a private key to PKCS#8 format using default parameters (AES with 256 bit key and hmacWithSHA256):

```
openssl pkcs8 -in key.pem -topk8 -out enckey.pem
```

Convert a private key to PKCS#8 unencrypted format:

```
openssl pkcs8 -in key.pem -topk8 -nocrypt -out enckey.pem
```

Convert a private key to PKCS#5 v2.0 format using triple DES:

```
openssl pkcs8 -in key.pem -topk8 -v2 des3 -out enckey.pem
```

Convert a private key to PKCS#5 v2.0 format using AES with 256 bits in CBC mode and hmacWithSHA512 PRF:

```
openssl pkcs8 -in key.pem -topk8 -v2 aes-256-cbc -v2prf hmacWithSHA512 -out enckey.pem
```

Convert a private key to PKCS#8 using a PKCS#5 1.5 compatible algorithm (DES):

```
openssl pkcs8 -in key.pem -topk8 -v1 PBE-MD5-DES -out enckey.pem
```

Convert a private key to PKCS#8 using a PKCS#12 compatible algorithm (3DES):

```
openssl pkcs8 -in key.pem -topk8 -out enckey.pem -v1 PBE-SHA1-3DES
```

Read a DER unencrypted PKCS#8 format private key:

```
openssl pkcs8 -inform DER -nocrypt -in key.der -out key.pem
```

Convert a private key from any PKCS#8 encrypted format to traditional format:

```
openssl pkcs8 -in pk8.pem -traditional -out key.pem
```

Convert a private key to PKCS#8 format, encrypting with AES-256 and with one million iterations of the password:

```
openssl pkcs8 -in key.pem -topk8 -v2 aes-256-cbc -iter 1000000 -out pk8.pem
```

STANDARDS

Test vectors from this PKCS#5 v2.0 implementation were posted to the pkcs-tng mailing list using triple DES, DES and RC2 with high iteration counts, several people confirmed that they could decrypt the private keys produced and therefore, it can be assumed that the PKCS#5 v2.0 implementation is reasonably accurate at least as far as these algorithms are concerned.

The format of PKCS#8 DSA (and other) private keys is not well documented: it is hidden away in PKCS#11 v2.01, section 11.9. OpenSSL's default DSA PKCS#8 private key format complies with this standard.

BUGS

There should be an option that prints out the encryption algorithm in use and other details such as the iteration count.

SEE ALSO

openssl(1), openssl-dsa(1), openssl-rsa(1), openssl-genrsa(1),
openssl-gendsa(1)

HISTORY

The -iter option was added in OpenSSL 1.1.0.

The -engine option was deprecated in OpenSSL 3.0.

COPYRIGHT

Copyright 2000-2021 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>>.

