## NAME

openssl-spkac, spkac - SPKAC printing and generating utility

## SYNOPSIS

**openssl spkac** [**-help**] [**-in filename**] [**-out filename**] [**-key keyfile**] [**-passin arg**] [**-challenge string**] [**-pubkey**] [**-spkac spkacname**] [**-spksect section**] [**-noout**] [**-verify**] [**-engine id**]

## DESCRIPTION

The **spkac** command processes Netscape signed public key and challenge (SPKAC) files. It can print out their contents, verify the signature and produce its own SPKACs from a supplied private key.

## OPTIONS

**-help**
Print out a usage message.

**-in filename**
This specifies the input filename to read from or standard input if this option is not specified. Ignored if the **-key** option is used.

**-out filename**
specifies the output filename to write to or standard output by default.

**-key keyfile**
create an SPKAC file using the private key in **keyfile**. The **-in**, **-noout**, **-spksect** and **-verify** options are ignored if present.

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

**-challenge string**
specifies the challenge string if an SPKAC is being created.

**-spkac spkacname**
allows an alternative name form the variable containing the SPKAC. The default is ''SPKAC''. This option affects both generated and input SPKAC files.

**-spksect section**
allows an alternative name form the section containing the SPKAC. The default is the default section.

**-noout**
don't output the text version of the SPKAC (not used if an SPKAC is being created).

**-pubkey**
output the public key of an SPKAC (not used if an SPKAC is being created).

**-verify**
verifies the digital signature on the supplied SPKAC.

**-engine id**
specifying an engine (by its unique **id** string) will cause **spkac** to attempt to obtain a functional reference to the specified engine, thus initialising it if needed. The engine will then be set as the default for all available algorithms.

## EXAMPLES

Print out the contents of an SPKAC:

```
 openssl spkac -in spkac.cnf
```

Verify the signature of an SPKAC:

```
 openssl spkac -in spkac.cnf -noout -verify
```

Create an SPKAC using the challenge string ''hello'':

```
 openssl spkac -key key.pem -challenge hello -out spkac.cnf
```

Example of an SPKAC, (long lines split up for clarity):

```
SPKAC=MIG5MGUwXDANBgkqhkiG9w0BAQEFAANLADBIAkEA1cCoq2Wa3Ixs47uI7F\
PVwHVIPDx5yso105Y6zpozam135a8R0CpoRvkkigIyXfcCjiVi5oWk+6FfPaD03u\
PFoQIDAQABFgVoZWxsbzANBgkqhkiG9w0BAQQFAANBAFpQtY/FojdwkJh1bEIYuc\
2EeM2KHTWPEepWYeawvHD0gQ3DngSC75YCWnnDdq+NQ3F+X4deMx9AaEglZtULwV\
4=
```

## NOTES

A created SPKAC with suitable DN components appended can be fed into the **ca** utility.

SPKACs are typically generated by Netscape when a form is submitted containing the **KEYGEN** tag as part of the certificate enrollment process.

The challenge string permits a primitive form of proof of possession of private key. By checking the SPKAC signature and a random challenge string some guarantee is given that the user knows the private key corresponding to the public key being certified. This is important in some applications. Without this it is possible for a previous SPKAC to be used in a ‘‘replay attack’’.

## SEE ALSO

*ca(1)*

## COPYRIGHT