NAME

openssl-pkeyutl - public key algorithm command

SYNOPSIS

openssl pkeyutl [**-help**] [**-in** *file*] [**-rawin**] [**-digest** *algorithm*] [**-out** *file*] [**-sigfile** *file*] [**-inkey** *filename*|*uri*] [**-keyform DER**|**PEM**|**P12**|**ENGINE**] [**-passin** *arg*] [**-peerkey** *file*] [**-peerform DER**|**PEM**|**P12**|**ENGINE**] [**-pubin**] [**-certin**] [**-rev**] [**-sign**] [**-verify**] [**-verifyrecover**] [**-encrypt**] [**-decrypt**] [**-derive**] [**-kdf** *algorithm*] [**-kdflen** *length*] [**-pkeyopt** *opt*:*value*] [**-pkeyopt_passin** *opt*[:*passarg*]] [**-hexdump**] [**-asn1parse**] [**-engine** *id*] [**-engine_impl**] [**-rand** *files*] [**-writerand** *file*] [**-provider** *name*] [**-provider-path** *path*] [**-propquery** *propq*] [**-config** *configfile*]

DESCRIPTION

This command can be used to perform low-level public key operations using any supported algorithm.

OPTIONS

-help

Print out a usage message.

-in *filename*

This specifies the input filename to read data from or standard input if this option is not specified.

-rawin

This indicates that the input data is raw data, which is not hashed by any message digest algorithm. The user can specify a digest algorithm by using the **-digest** option. This option can only be used with **-sign** and **-verify** and must be used with the Ed25519 and Ed448 algorithms.

-digest *algorithm*

This specifies the digest algorithm which is used to hash the input data before signing or verifying it with the input key. This option could be omitted if the signature algorithm does not require one (for instance, EdDSA). If this option is omitted but the signature algorithm requires one, a default value will be used. For signature algorithms like RSA, DSA and ECDSA, SHA-256 will be the default digest algorithm. For SM2, it will be SM3. If this option is present, then the **-rawin** option must be also specified.

-out *filename*

Specifies the output filename to write to or standard output by default.

-sigfile *file*

Signature file, required for **-verify** operations only

-inkey *filename*|*uri*
The input key, by default it should be a private key.

-keyform DER|PEM|P12|ENGINE
The key format; unspecified by default. See **openssl-format-options**(1) for details.

-passin *arg*
The input key password source. For more information about the format of *arg* see **openssl-passphrase-options**(1).

-peerkey *file*
The peer key file, used by key derivation (agreement) operations.

-peerform DER|PEM|P12|ENGINE
The peer key format; unspecified by default. See **openssl-format-options**(1) for details.

-pubin
The input file is a public key.

-certin
The input is a certificate containing a public key.

-rev Reverse the order of the input buffer. This is useful for some libraries (such as CryptoAPI) which represent the buffer in little endian format.

-sign
Sign the input data (which must be a hash) and output the signed result. This requires a private key.

-verify
Verify the input data (which must be a hash) against the signature file and indicate if the verification succeeded or failed.

-verifyrecover
Verify the input data (which must be a hash) and output the recovered data.

-encrypt
Encrypt the input data using a public key.

-decrypt

Decrypt the input data using a private key.

-derive
Derive a shared secret using the peer key.

-kdf *algorithm*
Use key derivation function *algorithm*. The supported algorithms are at present **TLS1-PRF** and **HKDF**. Note: additional parameters and the KDF output length will normally have to be set for this to work. See **EVP_PKEY_CTX_set_hkdf_md**(3) and **EVP_PKEY_CTX_set_tls1_prf_md**(3) for the supported string parameters of each algorithm.

-kdflen *length*
Set the output length for KDF.

-pkeyopt *opt*:*value*
Public key options specified as opt:value. See NOTES below for more details.

-pkeyopt_passin *opt*[:*passarg*]
Allows reading a public key option *opt* from stdin or a password source. If only *opt* is specified, the user will be prompted to enter a password on stdin. Alternatively, *passarg* can be specified which can be any value supported by **openssl-passphrase-options**(1).

-hexdump
hex dump the output data.

-asn1parse
Parse the ASN.1 output data, this is useful when combined with the **-verifyrecover** option when an ASN1 structure is signed.

-engine *id*
See "Engine Options" in **openssl**(1). This option is deprecated.

-engine_impl
When used with the **-engine** option, it specifies to also use engine *id* for crypto operations.

-rand *files,* **-writerand** *file*
See "Random State Options" in **openssl**(1) for details.

-provider *name*
-provider-path *path*

-propquery *propq*
See "Provider Options" in **openssl**(1), **provider**(7), and **property**(7).

-config *configfile*
See "Configuration Option" in **openssl**(1).

NOTES

The operations and options supported vary according to the key algorithm and its implementation. The OpenSSL operations and options are indicated below.

Unless otherwise mentioned all algorithms support the **digest:***alg* option which specifies the digest in use for sign, verify and verifyrecover operations. The value *alg* should represent a digest name as used in the **EVP_get_digestbyname()** function for example **sha1**. This value is not used to hash the input data. It is used (by some algorithms) for sanity-checking the lengths of data passed in and for creating the structures that make up the signature (e.g. **DigestInfo** in RSASSA PKCS#1 v1.5 signatures).

This command does not hash the input data (except where -rawin is used) but rather it will use the data directly as input to the signature algorithm. Depending on the key type, signature type, and mode of padding, the maximum acceptable lengths of input data differ. The signed data can't be longer than the key modulus with RSA. In case of ECDSA and DSA the data shouldn't be longer than the field size, otherwise it will be silently truncated to the field size. In any event the input size must not be larger than the largest supported digest size.

In other words, if the value of digest is **sha1** the input should be the 20 bytes long binary encoding of the SHA-1 hash function output.

RSA ALGORITHM

The RSA algorithm generally supports the encrypt, decrypt, sign, verify and verifyrecover operations. However, some padding modes support only a subset of these operations. The following additional **pkeyopt** values are supported:

rsa_padding_mode:*mode*
This sets the RSA padding mode. Acceptable values for *mode* are **pkcs1** for PKCS#1 padding, **none** for no padding, **oaep** for **OAEP** mode, **x931** for X9.31 mode and **pss** for PSS.

In PKCS#1 padding if the message digest is not set then the supplied data is signed or verified directly instead of using a **DigestInfo** structure. If a digest is set then the a **DigestInfo** structure is used and its the length must correspond to the digest type.

For **oaep** mode only encryption and decryption is supported.

For **x931** if the digest type is set it is used to format the block data otherwise the first byte is used to specify the X9.31 digest ID. Sign, verify and verifyrecover are can be performed in this mode.

For **pss** mode only sign and verify are supported and the digest type must be specified.

rsa_pss_saltlen:*len*

For **pss** mode only this option specifies the salt length. Three special values are supported: **digest** sets the salt length to the digest length, **max** sets the salt length to the maximum permissible value. When verifying **auto** causes the salt length to be automatically determined based on the **PSS** block structure.

rsa_mgf1_md:*digest*

For PSS and OAEP padding sets the MGF1 digest. If the MGF1 digest is not explicitly set in PSS mode then the signing digest is used.

rsa_oaep_md:*digest*

Sets the digest used for the OAEP hash function. If not explicitly set then SHA1 is used.

RSA-PSS ALGORITHM

The RSA-PSS algorithm is a restricted version of the RSA algorithm which only supports the sign and verify operations with PSS padding. The following additional **-pkeyopt** values are supported:

rsa_padding_mode:*mode*, **rsa_pss_saltlen:***len*, **rsa_mgf1_md:***digest*

These have the same meaning as the **RSA** algorithm with some additional restrictions. The padding mode can only be set to **pss** which is the default value.

If the key has parameter restrictions than the digest, MGF1 digest and salt length are set to the values specified in the parameters. The digest and MG cannot be changed and the salt length cannot be set to a value less than the minimum restriction.

DSA ALGORITHM

The DSA algorithm supports signing and verification operations only. Currently there are no additional **-pkeyopt** options other than **digest**. The SHA1 digest is assumed by default.

DH ALGORITHM

The DH algorithm only supports the derivation operation and no additional **-pkeyopt** options.

EC ALGORITHM

The EC algorithm supports sign, verify and derive operations. The sign and verify operations use ECDSA and derive uses ECDH. SHA1 is assumed by default for the **-pkeyopt digest** option.

X25519 AND X448 ALGORITHMS

The X25519 and X448 algorithms support key derivation only. Currently there are no additional options.

ED25519 AND ED448 ALGORITHMS

These algorithms only support signing and verifying. OpenSSL only implements the "pure" variants of these algorithms so raw data can be passed directly to them without hashing them first. The option **-rawin** must be used with these algorithms with no **-digest** specified. Additionally OpenSSL only supports "oneshot" operation with these algorithms. This means that the entire file to be signed/verified must be read into memory before processing it. Signing or Verifying very large files should be avoided. Additionally the size of the file must be known for this to work. If the size of the file cannot be determined (for example if the input is stdin) then the sign or verify operation will fail.

SM2

The SM2 algorithm supports sign, verify, encrypt and decrypt operations. For the sign and verify operations, SM2 requires an Distinguishing ID string to be passed in. The following **-pkeyopt** value is supported:

distid:*string*

This sets the ID string used in SM2 sign or verify operations. While verifying an SM2 signature, the ID string must be the same one used when signing the data. Otherwise the verification will fail.

hexdistid:*hex_string*

This sets the ID string used in SM2 sign or verify operations. While verifying an SM2 signature, the ID string must be the same one used when signing the data. Otherwise the verification will fail. The ID string provided with this option should be a valid hexadecimal value.

EXAMPLES

Sign some data using a private key:

```
 openssl pkeyutl -sign -in file -inkey key.pem -out sig
```

Recover the signed data (e.g. if an RSA key is used):

```
 openssl pkeyutl -verifyrecover -in sig -inkey key.pem
```

Verify the signature (e.g. a DSA key):

```
 openssl pkeyutl -verify -in file -sigfile sig -inkey key.pem
```

Sign data using a message digest value (this is currently only valid for RSA):

```
 openssl pkeyutl -sign -in file -inkey key.pem -out sig -pkeyopt digest:sha256
```

Derive a shared secret value:

```
 openssl pkeyutl -derive -inkey key.pem -peerkey pubkey.pem -out secret
```

Hexdump 48 bytes of TLS1 PRF using digest **SHA256** and shared secret and seed consisting of the single byte 0xFF:

```
 openssl pkeyutl -kdf TLS1-PRF -kdflen 48 -pkeyopt md:SHA256 \
    -pkeyopt hexsecret:ff -pkeyopt hexseed:ff -hexdump
```

Derive a key using **scrypt** where the password is read from command line:

```
 openssl pkeyutl -kdf scrypt -kdflen 16 -pkeyopt_passin pass \
    -pkeyopt hexsalt:aabbcc -pkeyopt N:16384 -pkeyopt r:8 -pkeyopt p:1
```

Derive using the same algorithm, but read key from environment variable MYPASS:

```
 openssl pkeyutl -kdf scrypt -kdflen 16 -pkeyopt_passin pass:env:MYPASS \
    -pkeyopt hexsalt:aabbcc -pkeyopt N:16384 -pkeyopt r:8 -pkeyopt p:1
```

Sign some data using an **SM2**(7) private key and a specific ID:

```
 openssl pkeyutl -sign -in file -inkey sm2.key -out sig -rawin -digest sm3 \
    -pkeyopt distid:someid
```

Verify some data using an **SM2**(7) certificate and a specific ID:

```
 openssl pkeyutl -verify -certin -in file -inkey sm2.cert -sigfile sig \
    -rawin -digest sm3 -pkeyopt distid:someid
```

Decrypt some data using a private key with OAEP padding using SHA256:

```
 openssl pkeyutl -decrypt -in file -inkey key.pem -out secret \
    -pkeyopt rsa_padding_mode:oaep -pkeyopt rsa_oaep_md:sha256
```

SEE ALSO

openssl(1), **openssl-genpkey**(1), **openssl-pkey**(1), **openssl-rsautl**(1) **openssl-dgst**(1), **openssl-rsa**(1), **openssl-genrsa**(1), **openssl-kdf**(1) **EVP_PKEY_CTX_set_hkdf_md**(3), **EVP_PKEY_CTX_set_tls1_prf_md**(3),

HISTORY

The **-engine** option was deprecated in OpenSSL 3.0.

COPYRIGHT