NAME

dgst, sha, sha1, mdc2, ripemd160, sha224, sha256, sha384, sha512, md2, md4, md5, dss1 - message digests

SYNOPSIS

openssl dgst [**-sha|-sha1|-mdc2|-ripemd160|-sha224|-sha256|-sha384|-sha512|-md2|-md4|-md5|-dss1**] [**-c**] [**-d**] [**-hex**] [**-binary**] [**-r**] [**-hmac arg**] [**-non-fips-allow**] [**-out filename**] [**-sign filename**] [**-keyform arg**] [**-passin arg**] [**-verify filename**] [**-prverify filename**] [**-signature filename**] [**-hmac key**] [**-non-fips-allow**] [**-fips-fingerprint**] [**file...**]

openssl [*digest*] [**...**]

DESCRIPTION

The digest functions output the message digest of a supplied file or files in hexadecimal. The digest functions also generate and verify digital signatures using message digests.

OPTIONS

-c print out the digest in two digit groups separated by colons, only relevant if **hex** format output is used.

-d print out BIO debugging information.

-hex
digest is to be output as a hex dump. This is the default case for a ‘‘normal’’ digest as opposed to a digital signature. See NOTES below for digital signatures using **-hex**.

-binary
output the digest or signature in binary form.

-r output the digest in the ‘‘coreutils’’ format used by programs like **sha1sum**.

-hmac arg
set the HMAC key to ‘‘arg’’.

-non-fips-allow
Allow use of non FIPS digest when in FIPS mode. This has no effect when not in FIPS mode.

-out filename
filename to output to, or standard output by default.

-sign filename
digitally sign the digest using the private key in ‘‘filename’’.

-keyform arg
Specifies the key format to sign digest with. The DER, PEM, P12, and ENGINE formats are supported.

-engine id
Use engine **id** for operations (including private key storage). This engine is not used as source for digest algorithms, unless it is also specified in the configuration file.

-sigopt nm:v
Pass options to the signature algorithm during sign or verify operations. Names and values of these options are algorithm-specific.

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

-verify filename
verify the signature using the the public key in ‘‘filename’’. The output is either ‘‘Verification OK’’ or ‘‘Verification Failure’’.

-prverify filename
verify the signature using the the private key in ‘‘filename’’.

-signature filename
the actual signature to verify.

-hmac key
create a hashed MAC using ‘‘key’’.

-mac alg
create MAC (keyed Message Authentication Code). The most popular MAC algorithm is HMAC (hash-based MAC), but there are other MAC algorithms which are not based on hash, for instance **gost-mac** algorithm, supported by **ccgost** engine. MAC keys and other options should be set via **-macopt** parameter.

-macopt nm:v
Passes options to MAC algorithm, specified by **-mac** key. Following options are supported by both by **HMAC** and **gost-mac**:

key:string
Specifies MAC key as alphnumeric string (use if key contain printable characters only). String length must conform to any restrictions of the MAC algorithm for example exactly 32 chars for gost-mac.

hexkey:string
Specifies MAC key in hexadecimal form (two hex digits per byte). Key length must conform to any restrictions of the MAC algorithm for example exactly 32 chars for gost-mac.

-rand file(s)
a file or files containing random data used to seed the random number generator, or an EGD socket (see *RAND_egd(3)*). Multiple files can be specified separated by a OS-dependent character. The separator is**;** for MS-Windo ws, **,** for OpenVMS, and **:** for all others.

-non-fips-allow
enable use of non-FIPS algorithms such as MD5 even in FIPS mode.

-fips-fingerprint
compute HMAC using a specific key for certain OpenSSL-FIPS operations.

file...
file or files to digest. If no files are specified then standard input is used.

EXAMPLES

To create a hex-encoded message digest of a file: openssl dgst -md5 -hex file.txt

To sign a file using SHA-256 with binary file output: openssl dgst -sha256 -sign privatekey.pem -out signature.sign file.txt

To verify a signature: openssl dgst -sha256 -verify publickey.pem -signature signature.sign file.txt

NOTES

The digest of choice for all new applications is SHA1. Other digests are however still widely used.

When signing a file, **dgst** will automatically determine the algorithm (RSA, ECC, etc) to use for signing based on the private key’s ASN.1 info. When verifying signatures, it only handles the RSA, DSA, or ECDSA signature itself, not the related data to identify the signer and algorithm used in formats such as x.509, CMS, and S/MIME.

A source of random numbers is required for certain signing algorithms, in particular ECDSA and DSA.

The signing and verify options should only be used if a single file is being signed or verified.

Hex signatures cannot be verified using **openssl**. Instead, use ‘‘xxd -r’’ or similar program to

transform the hex signature into a binary signature prior to verification.