NAME

ssh-keygen — OpenSSH authentication key utility

SYNOPSIS

ssh-keygen [-q] [-a *rounds*] [-b *bits*] [-C *comment*] [-f *output_keyfile*]
[-m *format*] [-N *new_passphrase*] [-O *option*]
[-t ecdsa | ecdsa-sk | ed25519 | ed25519-sk | rsa] [-w *provider*]
[-Z *cipher*]
ssh-keygen -p [-a *rounds*] [-f *keyfile*] [-m *format*] [-N *new_passphrase*]
[-P *old_passphrase*] [-Z *cipher*]
ssh-keygen -i [-f *input_keyfile*] [-m *key_format*]
ssh-keygen -e [-f *input_keyfile*] [-m *key_format*]
ssh-keygen -y [-f *input_keyfile*]
ssh-keygen -c [-a *rounds*] [-C *comment*] [-f *keyfile*] [-P *passphrase*]
ssh-keygen -l [-v] [-E *fingerprint_hash*] [-f *input_keyfile*]
ssh-keygen -B [-f *input_keyfile*]
ssh-keygen -D *pkcs11*
ssh-keygen -F *hostname* [-lv] [-f *known_hosts_file*]
ssh-keygen -H [-f *known_hosts_file*]
ssh-keygen -K [-a *rounds*] [-w *provider*]
ssh-keygen -R *hostname* [-f *known_hosts_file*]
ssh-keygen -r *hostname* [-g] [-f *input_keyfile*]
ssh-keygen -M generate [-O *option*] *output_file*
ssh-keygen -M screen [-f *input_file*] [-O *option*] *output_file*
ssh-keygen -I *certificate_identity* -s *ca_key* [-hU] [-D *pkcs11_provider*]
[-n *principals*] [-O *option*] [-V *validity_interval*]
[-z *serial_number*] *file ...*
ssh-keygen -L [-f *input_keyfile*]
ssh-keygen -A [-a *rounds*] [-f *prefix_path*]
ssh-keygen -k -f *krl_file* [-u] [-s *ca_public*] [-z *version_number*] *file ...*
ssh-keygen -Q [-l] -f *krl_file* *file ...*
ssh-keygen -Y find-principals [-O *option*] -s *signature_file* -f
allowed_signers_file
ssh-keygen -Y match-principals -I *signer_identity* -f
allowed_signers_file
ssh-keygen -Y check-novalidate [-O *option*] -n *namespace* -s *signature_file*
ssh-keygen -Y sign [-O *option*] -f *key_file* -n *namespace* *file ...*
ssh-keygen -Y verify [-O *option*] -f *allowed_signers_file* -I
signer_identity -n *namespace* -s *signature_file*
[-r *revocation_file*]

DESCRIPTION

ssh-keygen generates, manages and converts authentication keys for *ssh*(1). **ssh-keygen** can create keys for use by SSH protocol version 2.

The type of key to be generated is specified with the -t option. If invoked without any arguments, **ssh-keygen** will generate an Ed25519 key.

ssh-keygen is also used to generate groups for use in Diffie-Hellman group exchange (DH-GEX). See the "MODULI GENERATION" section for details.

Finally, **ssh-keygen** can be used to generate and update Key Revocation Lists, and to test whether given keys have been revoked by one. See the "KEY REVOCATION LISTS" section for details.

Normally each user wishing to use SSH with public key authentication runs this once to create the authentication key in *~/.ssh/id_ecdsa*, *~/.ssh/id_ecdsa_sk*, *~/.ssh/id_ed25519*, *~/.ssh/id_ed25519_sk* or *~/.ssh/id_rsa*. Additionally, the system administrator may use this to generate host keys.

Normally this program generates the key and asks for a file in which to store the private key. The public key is stored in a file with the same name but ".pub" appended. The program also asks for a passphrase. The passphrase may be empty to indicate no passphrase (host keys must have an empty passphrase), or it may be a string of arbitrary length. A passphrase is similar to a password, except it can be a phrase with a series of words, punctuation, numbers, whitespace, or any string of characters you want. Good passphrases are 10-30 characters long, are not simple sentences or otherwise easily guessable (English prose has only 1-2 bits of entropy per character, and provides very bad passphrases), and contain a mix of upper and lowercase letters, numbers, and non-alphanumeric characters. The passphrase can be changed later by using the `-p` option.

There is no way to recover a lost passphrase. If the passphrase is lost or forgotten, a new key must be generated and the corresponding public key copied to other machines.

ssh-keygen will by default write keys in an OpenSSH-specific format. This format is preferred as it offers better protection for keys at rest as well as allowing storage of key comments within the private key file itself. The key comment may be useful to help identify the key. The comment is initialized to "user@host" when the key is created, but can be changed using the `-c` option.

It is still possible for **ssh-keygen** to write the previously-used PEM format private keys using the `-m` flag. This may be used when generating new keys, and existing new-format keys may be converted using this option in conjunction with the `-p` (change passphrase) flag.

After a key is generated, **ssh-keygen** will ask where the keys should be placed to be activated.

The options are as follows:

`-A`
: Generate host keys of all default key types (rsa, ecdsa, and ed25519) if they do not already exist. The host keys are generated with the default key file path, an empty passphrase, default bits for the key type, and default comment. If `-f` has also been specified, its argument is used as a prefix to the default path for the resulting host key files. This is used by system administration scripts to generate new host keys.

`-a` *rounds*
: When saving a private key, this option specifies the number of KDF (key derivation function, currently *bcrypt_pbkdf*(3)) rounds used. Higher numbers result in slower passphrase verification and increased resistance to brute-force password cracking (should the keys be stolen). The default is 16 rounds.

`-B`
: Show the bubblebabble digest of specified private or public key file.

`-b` *bits*
: Specifies the number of bits in the key to create. For RSA keys, the minimum size is 1024 bits and the default is 3072 bits. Generally, 3072 bits is considered sufficient. For ECDSA keys, the `-b` flag determines the key length by selecting from one of three elliptic curve sizes: 256, 384 or 521 bits. Attempting to use bit lengths other than these three values for ECDSA keys will fail. ECDSA-SK, Ed25519 and Ed25519-SK keys have a fixed length and the `-b` flag will be ignored.

`-C` *comment*
: Provides a new comment.

`-c`
: Requests changing the comment in the private and public key files. The program will prompt for the file containing the private keys, for the passphrase if the key has one, and for the new comment.

`-D` *pkcs11*
: Download the public keys provided by the PKCS#11 shared library *pkcs11*. When used in combination with `-s`, this option indicates that a CA key resides in a PKCS#11 token (see the "CERTIFICATES" section for details).

`-E` *fingerprint_hash*
: Specifies the hash algorithm used when displaying key fingerprints. Valid options are: "md5" and "sha256". The default is "sha256".

`-e` This option will read a private or public OpenSSH key file and print to stdout a public key in one of the formats specified by the `-m` option. The default export format is "RFC4716". This option allows exporting OpenSSH keys for use by other programs, including several commercial SSH implementations.

`-F` *hostname* | *[hostname]:port*
Search for the specified *hostname* (with optional port number) in a *known_hosts* file, listing any occurrences found. This option is useful to find hashed host names or addresses and may also be used in conjunction with the `-H` option to print found keys in a hashed format.

`-f` *filename*
Specifies the filename of the key file.

`-g` Use generic DNS format when printing fingerprint resource records using the `-r` command.

`-H` Hash a *known_hosts* file. This replaces all hostnames and addresses with hashed representations within the specified file; the original content is moved to a file with a .old suffix. These hashes may be used normally by **ssh** and **sshd**, but they do not reveal identifying information should the file's contents be disclosed. This option will not modify existing hashed hostnames and is therefore safe to use on files that mix hashed and non-hashed names.

`-h` When signing a key, create a host certificate instead of a user certificate. See the "CERTIFICATES" section for details.

`-I` *certificate_identity*
Specify the key identity when signing a public key. See the "CERTIFICATES" section for details.

`-i` This option will read an unencrypted private (or public) key file in the format specified by the `-m` option and print an OpenSSH compatible private (or public) key to stdout. This option allows importing keys from other software, including several commercial SSH implementations. The default import format is "RFC4716".

`-K` Download resident keys from a FIDO authenticator. Public and private key files will be written to the current directory for each downloaded key. If multiple FIDO authenticators are attached, keys will be downloaded from the first touched authenticator. See the "FIDO AUTHENTICATOR" section for more information.

`-k` Generate a KRL file. In this mode, **ssh-keygen** will generate a KRL file at the location specified via the `-f` flag that revokes every key or certificate presented on the command line. Keys/certificates to be revoked may be specified by public key file or using the format described in the "KEY REVOCATION LISTS" section.

`-L` Prints the contents of one or more certificates.

`-l` Show fingerprint of specified public key file. **ssh-keygen** will try to find the matching public key file and prints its fingerprint. If combined with `-v`, a visual ASCII art representation of the key is supplied with the fingerprint.

`-M generate`
Generate candidate Diffie-Hellman Group Exchange (DH-GEX) parameters for eventual use by the 'diffie-hellman-group-exchange-*' key exchange methods. The numbers generated by this operation must be further screened before use. See the "MODULI GENERATION" section for more information.

`-M screen`
Screen candidate parameters for Diffie-Hellman Group Exchange. This will accept a list of candidate numbers and test that they are safe (Sophie Germain) primes with acceptable group generators. The results of this operation may be added to the */etc/ssh/moduli* file. See the "MODULI GENERATION" section for more information.

`-m` *key_format*
: Specify a key format for key generation, the `-i` (import), `-e` (export) conversion options, and the `-p` change passphrase operation. The latter may be used to convert between OpenSSH private key and PEM private key formats. The supported key formats are: "RFC4716" (RFC 4716/SSH2 public or private key), "PKCS8" (PKCS8 public or private key) or "PEM" (PEM public key). By default OpenSSH will write newly-generated private keys in its own format, but when converting public keys for export the default format is "RFC4716". Setting a format of "PEM" when generating or updating a supported private key type will cause the key to be stored in the legacy PEM private key format.

`-N` *new_passphrase*
: Provides the new passphrase.

`-n` *principals*
: Specify one or more principals (user or host names) to be included in a certificate when signing a key. Multiple principals may be specified, separated by commas. See the "CERTIFICATES" section for details.

`-O` *option*
: Specify a key/value option. These are specific to the operation that **ssh-keygen** has been requested to perform.

  When signing certificates, one of the options listed in the "CERTIFICATES" section may be specified here.

  When performing moduli generation or screening, one of the options listed in the "MODULI GENERATION" section may be specified.

  When generating FIDO authenticator-backed keys, the options listed in the "FIDO AUTHENTICATOR" section may be specified.

  When performing signature-related options using the `-Y` flag, the following options are accepted:

  `hashalg=`*algorithm*
  : Selects the hash algorithm to use for hashing the message to be signed. Valid algorithms are "sha256" and "sha512." The default is "sha512."

  `print-pubkey`
  : Print the full public key to standard output after signature verification.

  `verify-time=`*timestamp*
  : Specifies a time to use when validating signatures instead of the current time. The time may be specified as a date or time in the YYYYMMDD[Z] or in YYYYMMDDHHMM[SS][Z] formats. Dates and times will be interpreted in the current system time zone unless suffixed with a Z character, which causes them to be interpreted in the UTC time zone.

  When generating SSHFP DNS records from public keys using the `-r` flag, the following options are accepted:

  `hashalg=`*algorithm*
  : Selects a hash algorithm to use when printing SSHFP records using the `-D` flag. Valid algorithms are "sha1" and "sha256". The default is to print both.

  The `-O` option may be specified multiple times.

`-P` *passphrase*
: Provides the (old) passphrase.

`-p`
: Requests changing the passphrase of a private key file instead of creating a new private key. The program will prompt for the file containing the private key, for the old passphrase, and twice for the new passphrase.

`-Q` Test whether keys have been revoked in a KRL. If the `-l` option is also specified then the contents of the KRL will be printed.

`-q` Silence **ssh-keygen**.

`-R` *hostname* | *[hostname]:port*
Removes all keys belonging to the specified *hostname* (with optional port number) from a *known_hosts* file. This option is useful to delete hashed hosts (see the `-H` option above).

`-r` *hostname*
Print the SSHFP fingerprint resource record named *hostname* for the specified public key file.

`-s` *ca_key*
Certify (sign) a public key using the specified CA key. See the "CERTIFICATES" section for details.

When generating a KRL, `-s` specifies a path to a CA public key file used to revoke certificates directly by key ID or serial number. See the "KEY REVOCATION LISTS" section for details.

`-t ecdsa | ecdsa-sk | ed25519 | ed25519-sk | rsa`
Specifies the type of key to create. The possible values are "ecdsa", "ecdsa-sk", "ed25519 (the default)," "ed25519-sk", or "rsa".

This flag may also be used to specify the desired signature type when signing certificates using an RSA CA key. The available RSA signature variants are "ssh-rsa" (SHA1 signatures, not recommended), "rsa-sha2-256", and "rsa-sha2-512" (the default for RSA keys).

`-U` When used in combination with `-s` or `-Y sign`, this option indicates that a CA key resides in an *ssh-agent*(1). See the "CERTIFICATES" section for more information.

`-u` Update a KRL. When specified with `-k`, keys listed via the command line are added to the existing KRL rather than a new KRL being created.

`-V` *validity_interval*
Specify a validity interval when signing a certificate. A validity interval may consist of a single time, indicating that the certificate is valid beginning now and expiring at that time, or may consist of two times separated by a colon to indicate an explicit time interval.

The start time may be specified as:
- The string "always" to indicate the certificate has no specified start time.
- A date or time in the system time zone formatted as YYYYMMDD or YYYYMMDDHHMM[SS].
- A date or time in the UTC time zone as YYYYMMDDZ or YYYYMMDDHHMM[SS]Z.
- A relative time before the current system time consisting of a minus sign followed by an interval in the format described in the TIME FORMATS section of *sshd_config*(5).
- A raw seconds since epoch (Jan 1 1970 00:00:00 UTC) as a hexadecimal number beginning with "0x".

The end time may be specified similarly to the start time:
- The string "forever" to indicate the certificate has no specified end time.
- A date or time in the system time zone formatted as YYYYMMDD or YYYYMMDDHHMM[SS].
- A date or time in the UTC time zone as YYYYMMDDZ or YYYYMMDDHHMM[SS]Z.
- A relative time after the current system time consisting of a plus sign followed by an interval in the format described in the TIME FORMATS section of *sshd_config*(5).
- A raw seconds since epoch (Jan 1 1970 00:00:00 UTC) as a hexadecimal number beginning with "0x".

For example:

+52w1d
: Valid from now to 52 weeks and one day from now.

-4w:+4w
: Valid from four weeks ago to four weeks from now.

20100101123000:20110101123000
: Valid from 12:30 PM, January 1st, 2010 to 12:30 PM, January 1st, 2011.

20100101123000Z:20110101123000Z
: Similar, but interpreted in the UTC time zone rather than the system time zone.

-1d:20110101
: Valid from yesterday to midnight, January 1st, 2011.

0x1:0x2000000000
: Valid from roughly early 1970 to May 2033.

-1m:forever
: Valid from one minute ago and never expiring.

`-v` Verbose mode. Causes **ssh-keygen** to print debugging messages about its progress. This is helpful for debugging moduli generation. Multiple `-v` options increase the verbosity. The maximum is 3.

`-w` *provider*
: Specifies a path to a library that will be used when creating FIDO authenticator-hosted keys, overriding the default of using the internal USB HID support.

`-Y find-principals`
: Find the principal(s) associated with the public key of a signature, provided using the `-s` flag in an authorized signers file provided using the `-f` flag. The format of the allowed signers file is documented in the "ALLOWED SIGNERS" section below. If one or more matching principals are found, they are returned on standard output.

`-Y match-principals`
: Find principal matching the principal name provided using the `-I` flag in the authorized signers file specified using the `-f` flag. If one or more matching principals are found, they are returned on standard output.

`-Y check-novalidate`
: Checks that a signature generated using **ssh-keygen** `-Y sign` has a valid structure. This does not validate if a signature comes from an authorized signer. When testing a signature, **ssh-keygen** accepts a message on standard input and a signature namespace using `-n`. A file containing the corresponding signature must also be supplied using the `-s` flag. Successful testing of the signature is signalled by **ssh-keygen** returning a zero exit status.

`-Y sign`
: Cryptographically sign a file or some data using an SSH key. When signing, **ssh-keygen** accepts zero or more files to sign on the command-line - if no files are specified then **ssh-keygen** will sign data presented on standard input. Signatures are written to the path of the input file with ".sig" appended, or to standard output if the message to be signed was read from standard input.

  The key used for signing is specified using the `-f` option and may refer to either a private key, or a public key with the private half available via *ssh-agent*(1). An additional signature namespace, used to prevent signature confusion across different domains of use (e.g. file signing vs email signing) must be provided via the `-n` flag. Namespaces are arbitrary strings, and may include: "file" for file signing, "email" for email signing. For custom uses, it is recommended to use names following a NAMESPACE@YOUR.DOMAIN pattern to generate unambiguous namespaces.

`-Y verify`
: Request to verify a signature generated using **ssh-keygen** `-Y sign` as described above. When verifying a signature, **ssh-keygen** accepts a message on standard input and a signature namespace using `-n`. A file containing the corresponding signature must also be supplied using the `-s` flag, along with the identity of the signer using `-I` and a list of allowed signers via the `-f` flag. The format of the allowed signers file is documented in the "ALLOWED SIGNERS" section below. A file containing revoked keys can be passed using the `-r` flag. The revocation file may be a KRL or a one-per-line list of public keys. Successful verification by an authorized signer is signalled by **ssh-keygen** returning a zero exit status.

`-y`
: This option will read a private OpenSSH format file and print an OpenSSH public key to stdout.

`-Z` *cipher*
: Specifies the cipher to use for encryption when writing an OpenSSH-format private key file. The list of available ciphers may be obtained using "ssh -Q cipher". The default is "aes256-ctr".

`-z` *serial_number*
: Specifies a serial number to be embedded in the certificate to distinguish this certificate from others from the same CA. If the *serial_number* is prefixed with a '+' character, then the serial number will be incremented for each certificate signed on a single command-line. The default serial number is zero.

  When generating a KRL, the `-z` flag is used to specify a KRL version number.

MODULI GENERATION

ssh-keygen may be used to generate groups for the Diffie-Hellman Group Exchange (DH-GEX) protocol. Generating these groups is a two-step process: first, candidate primes are generated using a fast, but memory intensive process. These candidate primes are then tested for suitability (a CPU-intensive process).

Generation of primes is performed using the `-M generate` option. The desired length of the primes may be specified by the `-O bits` option. For example:

```
# ssh-keygen -M generate -O bits=2048 moduli-2048.candidates
```

By default, the search for primes begins at a random point in the desired length range. This may be overridden using the `-O start` option, which specifies a different start point (in hex).

Once a set of candidates have been generated, they must be screened for suitability. This may be performed using the `-M screen` option. In this mode **ssh-keygen** will read candidates from standard input (or a file specified using the `-f` option). For example:

```
# ssh-keygen -M screen -f moduli-2048.candidates moduli-2048
```

By default, each candidate will be subjected to 100 primality tests. This may be overridden using the `-O prime-tests` option. The DH generator value will be chosen automatically for the prime under consideration. If a specific generator is desired, it may be requested using the `-O generator` option. Valid generator values are 2, 3, and 5.

Screened DH groups may be installed in */etc/ssh/moduli*. It is important that this file contains moduli of a range of bit lengths.

A number of options are available for moduli generation and screening via the `-O` flag:

lines=*number*
: Exit after screening the specified number of lines while performing DH candidate screening.

start-line=*line-number*
: Start screening at the specified line number while performing DH candidate screening.

checkpoint=*filename*
: Write the last line processed to the specified file while performing DH candidate screening. This will be used to skip lines in the input file that have already been processed if the job is restarted.

start=*hex-value*
Specify start point (in hex) when generating candidate moduli for DH-GEX.

generator=*value*
Specify desired generator (in decimal) when testing candidate moduli for DH-GEX.

CERTIFICATES

ssh-keygen supports signing of keys to produce certificates that may be used for user or host authentication. Certificates consist of a public key, some identity information, zero or more principal (user or host) names and a set of options that are signed by a Certification Authority (CA) key. Clients or servers may then trust only the CA key and verify its signature on a certificate rather than trusting many user/host keys. Note that OpenSSH certificates are a different, and much simpler, format to the X.509 certificates used in *ssl*(8).

ssh-keygen supports two types of certificates: user and host. User certificates authenticate users to servers, whereas host certificates authenticate server hosts to users. To generate a user certificate:

```
$ ssh-keygen -s /path/to/ca_key -I key_id /path/to/user_key.pub
```

The resultant certificate will be placed in */path/to/user_key-cert.pub*. A host certificate requires the `-h` option:

```
$ ssh-keygen -s /path/to/ca_key -I key_id -h /path/to/host_key.pub
```

The host certificate will be output to */path/to/host_key-cert.pub*.

It is possible to sign using a CA key stored in a PKCS#11 token by providing the token library using `-D` and identifying the CA key by providing its public half as an argument to `-s`:

```
$ ssh-keygen -s ca_key.pub -D libpkcs11.so -I key_id user_key.pub
```

Similarly, it is possible for the CA key to be hosted in an *ssh-agent*(1). This is indicated by the `-U` flag and, again, the CA key must be identified by its public half.

```
$ ssh-keygen -Us ca_key.pub -I key_id user_key.pub
```

In all cases, *key_id* is a "key identifier" that is logged by the server when the certificate is used for authentication.

Certificates may be limited to be valid for a set of principal (user/host) names. By default, generated certificates are valid for all users or hosts. To generate a certificate for a specified set of principals:

```
$ ssh-keygen -s ca_key -I key_id -n user1,user2 user_key.pub
$ ssh-keygen -s ca_key -I key_id -h -n host.domain host_key.pub
```

Additional limitations on the validity and use of user certificates may be specified through certificate options. A certificate option may disable features of the SSH session, may be valid only when presented from particular source addresses or may force the use of a specific command.

The options that are valid for user certificates are:

clear Clear all enabled permissions. This is useful for clearing the default set of permissions so permissions may be added individually.

critical:*name*[=*contents*]
extension:*name*[=*contents*]
Includes an arbitrary certificate critical option or extension. The specified *name* should include a domain suffix, e.g. “name@example.com”. If *contents* is specified then it is included as the contents of the extension/option encoded as a string, otherwise the extension/option is created with no contents (usually indicating a flag). Extensions may be ignored by a client or server that does not recognise them, whereas unknown critical options will cause the certificate to be refused.

force-command=*command*
Forces the execution of *command* instead of any shell or command specified by the user when the certificate is used for authentication.

no-agent-forwarding
Disable *ssh-agent*(1) forwarding (permitted by default).

no-port-forwarding
Disable port forwarding (permitted by default).

no-pty
Disable PTY allocation (permitted by default).

no-user-rc
Disable execution of *~/.ssh/rc* by *sshd*(8) (permitted by default).

no-x11-forwarding
Disable X11 forwarding (permitted by default).

permit-agent-forwarding
Allows *ssh-agent*(1) forwarding.

permit-port-forwarding
Allows port forwarding.

permit-pty
Allows PTY allocation.

permit-user-rc
Allows execution of *~/.ssh/rc* by *sshd*(8).

permit-X11-forwarding
Allows X11 forwarding.

no-touch-required
Do not require signatures made using this key include demonstration of user presence (e.g. by having the user touch the authenticator). This option only makes sense for the FIDO authenticator algorithms `ecdsa-sk` and `ed25519-sk`.

source-address=*address_list*
Restrict the source addresses from which the certificate is considered valid. The *address_list* is a comma-separated list of one or more address/netmask pairs in CIDR format.

verify-required
Require signatures made using this key indicate that the user was first verified, e.g. by PIN or on-token biometrics. This option only makes sense for the FIDO authenticator algorithms `ecdsa-sk` and `ed25519-sk`.

At present, no standard options are valid for host keys.

Finally, certificates may be defined with a validity lifetime. The `-V` option allows specification of certificate start and end times. A certificate that is presented at a time outside this range will not be considered valid. By default, certificates are valid from the Unix Epoch to the distant future.

For certificates to be used for user or host authentication, the CA public key must be trusted by *sshd*(8) or *ssh*(1). Refer to those manual pages for details.

FIDO AUTHENTICATOR

ssh-keygen is able to generate FIDO authenticator-backed keys, after which they may be used much like any other key type supported by OpenSSH, so long as the hardware authenticator is attached when the keys are used. FIDO authenticators generally require the user to explicitly authorise operations by touching or tapping them. FIDO keys consist of two parts: a key handle part stored in the private key file on disk, and a per-device private key that is unique to each FIDO authenticator and that cannot be exported from the authenticator hardware. These are combined by the hardware at authentication time to derive the real key that is used to sign authentication challenges. Supported key types are `ecdsa-sk` and `ed25519-sk`.

The options that are valid for FIDO keys are:

`application`
: Override the default FIDO application/origin string of "ssh:". This may be useful when generating host or domain-specific resident keys. The specified application string must begin with "ssh:".

`challenge=`*`path`*
: Specifies a path to a challenge string that will be passed to the FIDO authenticator during key generation. The challenge string may be used as part of an out-of-band protocol for key enrollment (a random challenge is used by default).

`device`
: Explicitly specify a *fido*(4) device to use, rather than letting the authenticator middleware select one.

`no-touch-required`
: Indicate that the generated private key should not require touch events (user presence) when making signatures. Note that *sshd*(8) will refuse such signatures by default, unless overridden via an authorized_keys option.

`resident`
: Indicate that the key handle should be stored on the FIDO authenticator itself. This makes it easier to use the authenticator on multiple computers. Resident keys may be supported on FIDO2 authenticators and typically require that a PIN be set on the authenticator prior to generation. Resident keys may be loaded off the authenticator using *ssh-add*(1). Storing both parts of a key on a FIDO authenticator increases the likelihood of an attacker being able to use a stolen authenticator device.

`user` A username to be associated with a resident key, overriding the empty default username. Specifying a username may be useful when generating multiple resident keys for the same application name.

`verify-required`
: Indicate that this private key should require user verification for each signature. Not all FIDO authenticators support this option. Currently PIN authentication is the only supported verification method, but other methods may be supported in the future.

`write-attestation=`*`path`*
: May be used at key generation time to record the attestation data returned from FIDO authenticators during key generation. This information is potentially sensitive. By default, this information is discarded.

KEY REVOCATION LISTS

`ssh-keygen` is able to manage OpenSSH format Key Revocation Lists (KRLs). These binary files specify keys or certificates to be revoked using a compact format, taking as little as one bit per certificate if they are being revoked by serial number.

KRLs may be generated using the `-k` flag. This option reads one or more files from the command line and generates a new KRL. The files may either contain a KRL specification (see below) or public keys, listed one per line. Plain public keys are revoked by listing their hash or contents in the KRL and certificates revoked by serial number or key ID (if the serial is zero or not available).

Revoking keys using a KRL specification offers explicit control over the types of record used to revoke keys and may be used to directly revoke certificates by serial number or key ID without having the complete original certificate on hand. A KRL specification consists of lines containing one of the following directives followed by a colon and some directive-specific information.

`serial:` *`serial_number`*[`-`*`serial_number`*]
: Revokes a certificate with the specified serial number. Serial numbers are 64-bit values, not including zero and may be expressed in decimal, hex or octal. If two serial numbers are specified separated by a hyphen, then the range of serial numbers including and between each is revoked. The CA key must have been specified on the **`ssh-keygen`** command line using the `-s` option.

id: *key_id*
: Revokes a certificate with the specified key ID string. The CA key must have been specified on the **ssh-keygen** command line using the `-s` option.

key: *public_key*
: Revokes the specified key. If a certificate is listed, then it is revoked as a plain public key.

sha1: *public_key*
: Revokes the specified key by including its SHA1 hash in the KRL.

sha256: *public_key*
: Revokes the specified key by including its SHA256 hash in the KRL. KRLs that revoke keys by SHA256 hash are not supported by OpenSSH versions prior to 7.9.

hash: *fingerprint*
: Revokes a key using a fingerprint hash, as obtained from an *sshd*(8) authentication log message or the **ssh-keygen** `-l` flag. Only SHA256 fingerprints are supported here and resultant KRLs are not supported by OpenSSH versions prior to 7.9.

KRLs may be updated using the `-u` flag in addition to `-k`. When this option is specified, keys listed via the command line are merged into the KRL, adding to those already there.

It is also possible, given a KRL, to test whether it revokes a particular key (or keys). The `-Q` flag will query an existing KRL, testing each key specified on the command line. If any key listed on the command line has been revoked (or an error encountered) then **ssh-keygen** will exit with a non-zero exit status. A zero exit status will only be returned if no key was revoked.

ALLOWED SIGNERS

When verifying signatures, **ssh-keygen** uses a simple list of identities and keys to determine whether a signature comes from an authorized source. This "allowed signers" file uses a format patterned after the AUTHORIZED_KEYS FILE FORMAT described in *sshd*(8). Each line of the file contains the following space-separated fields: principals, options, keytype, base64-encoded key. Empty lines and lines starting with a '`#`' are ignored as comments.

The principals field is a pattern-list (see PATTERNS in *ssh_config*(5)) consisting of one or more comma-separated USER@DOMAIN identity patterns that are accepted for signing. When verifying, the identity presented via the `-I` option must match a principals pattern in order for the corresponding key to be considered acceptable for verification.

The options (if present) consist of comma-separated option specifications. No spaces are permitted, except within double quotes. The following option specifications are supported (note that option keywords are case-insensitive):

`cert-authority`
: Indicates that this key is accepted as a certificate authority (CA) and that certificates signed by this CA may be accepted for verification.

`namespaces`=namespace-list
: Specifies a pattern-list of namespaces that are accepted for this key. If this option is present, the signature namespace embedded in the signature object and presented on the verification command-line must match the specified list before the key will be considered acceptable.

`valid-after`=timestamp
: Indicates that the key is valid for use at or after the specified timestamp, which may be a date or time in the YYYYMMDD[Z] or YYYYMMDDHHMM[SS][Z] formats. Dates and times will be interpreted in the current system time zone unless suffixed with a Z character, which causes them to be interpreted in the UTC time zone.

`valid-before`=timestamp
: Indicates that the key is valid for use at or before the specified timestamp.

When verifying signatures made by certificates, the expected principal name must match both the principals pattern in the allowed signers file and the principals embedded in the certificate itself.

An example allowed signers file:

```
# Comments allowed at start of line
user1@example.com,user2@example.com ssh-rsa AAAAX1...
# A certificate authority, trusted for all principals in a domain.
*@example.com cert-authority ssh-ed25519 AAAB4...
# A key that is accepted only for file signing.
user2@example.com namespaces="file" ssh-ed25519 AAA41...
```

ENVIRONMENT

`SSH_SK_PROVIDER`
: Specifies a path to a library that will be used when loading any FIDO authenticator-hosted keys, overriding the default of using the built-in USB HID support.

FILES

~/.ssh/id_ecdsa
~/.ssh/id_ecdsa_sk
~/.ssh/id_ed25519
~/.ssh/id_ed25519_sk
~/.ssh/id_rsa
: Contains the ECDSA, authenticator-hosted ECDSA, Ed25519, authenticator-hosted Ed25519 or RSA authentication identity of the user. This file should not be readable by anyone but the user. It is possible to specify a passphrase when generating the key; that passphrase will be used to encrypt the private part of this file using 128-bit AES. This file is not automatically accessed by **ssh-keygen** but it is offered as the default file for the private key. *ssh*(1) will read this file when a login attempt is made.

~/.ssh/id_ecdsa.pub
~/.ssh/id_ecdsa_sk.pub
~/.ssh/id_ed25519.pub
~/.ssh/id_ed25519_sk.pub
~/.ssh/id_rsa.pub
: Contains the ECDSA, authenticator-hosted ECDSA, Ed25519, authenticator-hosted Ed25519 or RSA public key for authentication. The contents of this file should be added to *~/.ssh/authorized_keys* on all machines where the user wishes to log in using public key authentication. There is no need to keep the contents of this file secret.

/etc/ssh/moduli
: Contains Diffie-Hellman groups used for DH-GEX. The file format is described in *moduli*(5).

SEE ALSO

ssh(1), *ssh-add*(1), *ssh-agent*(1), *moduli*(5), *sshd*(8)

The Secure Shell (SSH) Public Key File Format, RFC 4716, 2006.

AUTHORS

OpenSSH is a derivative of the original and free ssh 1.2.12 release by Tatu Ylonen. Aaron Campbell, Bob Beck, Markus Friedl, Niels Provos, Theo de Raadt and Dug Song removed many bugs, re-added newer features and created OpenSSH. Markus Friedl contributed the support for SSH protocol versions 1.5 and 2.0.