NAME

ssh-keygen — authentication key generation, management and conversion

SYNOPSIS

ssh-keygen [**-q**] [**-b** *bits*] [**-t dsa** | **ecdsa** | **ed25519** | **rsa**] [**-N** *new_passphrase*]
[**-C** *comment*] [**-f** *output_keyfile*]
ssh-keygen -p [**-P** *old_passphrase*] [**-N** *new_passphrase*] [**-f** *keyfile*]
ssh-keygen -i [**-m** *key_format*] [**-f** *input_keyfile*]
ssh-keygen -e [**-m** *key_format*] [**-f** *input_keyfile*]
ssh-keygen -y [**-f** *input_keyfile*]
ssh-keygen -c [**-P** *passphrase*] [**-C** *comment*] [**-f** *keyfile*]
ssh-keygen -l [**-v**] [**-E** *fingerprint_hash*] [**-f** *input_keyfile*]
ssh-keygen -B [**-f** *input_keyfile*]
ssh-keygen -D *pkcs11*
ssh-keygen -F *hostname* [**-f** *known_hosts_file*] [**-l**]
ssh-keygen -H [**-f** *known_hosts_file*]
ssh-keygen -R *hostname* [**-f** *known_hosts_file*]
ssh-keygen -r *hostname* [**-f** *input_keyfile*] [**-g**]
ssh-keygen -G *output_file* [**-v**] [**-b** *bits*] [**-M** *memory*] [**-S** *start_point*]
ssh-keygen -T *output_file* **-f** *input_file* [**-v**] [**-a** *rounds*] [**-J** *num_lines*]
[**-j** *start_line*] [**-K** *checkpt*] [**-W** *generator*]
ssh-keygen -s *ca_key* **-I** *certificate_identity* [**-h**] [**-U**] [**-D** *pkcs11_provider*]
[**-n** *principals*] [**-O** *option*] [**-V** *validity_interval*]
[**-z** *serial_number*] *file ...*
ssh-keygen -L [**-f** *input_keyfile*]
ssh-keygen -A [**-f** *prefix_path*]
ssh-keygen -k -f *krl_file* [**-u**] [**-s** *ca_public*] [**-z** *version_number*] *file ...*
ssh-keygen -Q -f *krl_file 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 RSA 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Finally, **ssh-keygen** can be used to generate and update Key Revocation Lists, and to test whether given keys have been revoked by one.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_dsa, ~/.ssh/id_ecdsa, ~/.ssh/id_ed25519 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.

For keys stored in the newer OpenSSH format, there is also a comment field in the key file that is only for convenience to the user to help identify the key. The comment can tell what the key is for, or whatever is useful. The comment is initialized to "user@host" when the key is created, but can be changed using the **-c** option.

After a key is generated, instructions below detail where the keys should be placed to be activated.

The options are as follows:

-A For each of the key types (rsa, dsa, ecdsa and ed25519) for which host keys do not exist, generate the host keys 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 new-format private key (i.e. an ed25519 key or when the **-o** flag is set), this option specifies the number of KDF (key derivation function) rounds used. Higher numbers result in slower passphrase verification and increased resistance to brute-force password cracking (should the keys be stolen).

When screening DH-GEX candidates (using the **-T** command). This option specifies the number of primality tests to perform.

-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 2048 bits. Generally, 2048 bits is considered sufficient. DSA keys must be exactly 1024 bits as specified by FIPS 186-2. 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. Ed25519 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. This operation is only supported for keys stored in the newer OpenSSH format. 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 RSA 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 the 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*
Search for the specified *hostname* 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 *output_file*
Generate candidate primes for DH-GEX. These primes must be screened for safety (using the **-T** option) before use.

-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. Please see the **CERTIFICATES** section for details.

-I *certificate_identity*
Specify the key identity when signing a public key. Please 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".

-J *num_lines*
Exit after screening the specified number of lines while performing DH candidate screening using the **-T** option.

-j *start_line*
Start screening at the specified line number while performing DH candidate screening using the **-T** option.

-K *checkpt*
Write the last line processed to the file *checkpt* while performing DH candidate screening using the **-T** option. This will be used to skip lines in the input file that have already been processed if the job is restarted.

-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. For RSA and DSA keys **ssh-keygen** tries 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 *memory*
Specify the amount of memory to use (in megabytes) when generating candidate moduli for DH-GEX.

-m *key_format*
Specify a key format for the **-i** (import) or **-e** (export) conversion options. The supported key formats are: "RFC4716" (RFC 4716/SSH2 public or private key), "PKCS8" (PEM PKCS8 public key) or "PEM" (PEM public key). The default conversion format is "RFC4716".

-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. Please see the **CERTIFICATES** section for details.

-O *option*
Specify a certificate option when signing a key. This option may be specified multiple times. See also the **CERTIFICATES** section for further details. 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.

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

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.

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.

-o Causes **ssh-keygen** to save private keys using the new OpenSSH format rather than the more compatible PEM format. The new format has increased resistance to brute-force password cracking but is not supported by versions of OpenSSH prior to 6.5. Ed25519 keys always use the new private key format.

-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.

-q Silence **ssh-keygen**.

-R *hostname*
Removes all keys belonging to *hostname* 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 *start*
Specify start point (in hex) when generating candidate moduli for DH-GEX.

-s *ca_key*
Certify (sign) a public key using the specified CA key. Please 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 *output_file*
Test DH group exchange candidate primes (generated using the **-G** option) for safety.

-t dsa | **ecdsa** | **ed25519** | **rsa**
Specifies the type of key to create. The possible values are “dsa”, “ecdsa”, “ed25519”, or “rsa”.

-U When used in combination with **-s**, this option indicates that a CA key resides in a 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 a date in YYYYMMDD format, a time in YYYYMMDDHHMMSS format or a relative time (to the current time) consisting of a minus sign followed by a relative time in the format described in the TIME FORMATS section of `sshd_config`(5). The end time may be specified as a YYYYMMDD date, a YYYYMMDDHHMMSS time or a relative time starting with a plus character.

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), "-1d:20110101" (valid from yesterday to midnight, January 1st, 2011).

-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 *generator*
Specify desired generator when testing candidate moduli for DH-GEX.

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

-z *serial_number*
Specifies a serial number to be embedded in the certificate to distinguish this certificate from others from the same CA. 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 **-G** option. The desired length of the primes may be specified by the **-b** option. For example:

```
# ssh-keygen -G moduli-2048.candidates -b 2048
```

By default, the search for primes begins at a random point in the desired length range. This may be overridden using the **-S** 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 **-T** option. In this mode **ssh-keygen** will read candidates from standard input (or a file specified using the **-f** option). For example:

```
# ssh-keygen -T moduli-2048 -f moduli-2048.candidates
```

By default, each candidate will be subjected to 100 primality tests. This may be overridden using the **-a** 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 **-W** 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 and that both ends of a connection share common moduli.

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 a 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. For a list of valid certificate options, see the documentation for the **-O** option above.

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 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). Please refer to those manual pages for details.

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 fol-

lowed 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 its SHA1 hash.

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.

FILES

`~/.ssh/id_dsa`
`~/.ssh/id_ecdsa`
`~/.ssh/id_ed25519`
`~/.ssh/id_rsa`
: Contains the DSA, ECDSA, 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_dsa.pub`
`~/.ssh/id_ecdsa.pub`
`~/.ssh/id_ed25519.pub`
`~/.ssh/id_rsa.pub`
: Contains the DSA, ECDSA, 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 fea-

tures and created OpenSSH. Markus Friedl contributed the support for SSH protocol versions 1.5 and 2.0.