NAME

ntp-keygen - create a Network Time Protocol host key

SYNOPSIS

ntp-keygen [**-flags**] [**-flag** [*value*]] [**--option-name**[[=|]*value*]]

All arguments must be options.

DESCRIPTION

This program generates cryptographic data files used by the NTPv4 authentication and identification schemes. It can generate message digest keys used in symmetric key cryptography and, if the OpenSSL software library has been installed, it can generate host keys, signing keys, certificates, and identity keys and parameters used in Autokey public key cryptography. These files are used for cookie encryption, digital signature, and challenge/response identification algorithms compatible with the Internet standard security infrastructure.

The message digest symmetric keys file is generated in a format compatible with NTPv3. All other files are in PEM-encoded printable ASCII format, so they can be embedded as MIME attachments in email to other sites and certificate authorities. By default, files are not encrypted.

When used to generate message digest symmetric keys, the program produces a file containing ten pseudo-random printable ASCII strings suitable for the MD5 message digest algorithm included in the distribution. If the OpenSSL library is installed, it produces an additional ten hex-encoded random bit strings suitable for SHA1, AES-128-CMAC, and other message digest algorithms. The message digest symmetric keys file must be distributed and stored using secure means beyond the scope of NTP itself. Besides the keys used for ordinary NTP associations, additional keys can be defined as passwords for the ntpq(8) and ntpdc(8) utility programs.

The remaining generated files are compatible with other OpenSSL applications and other Public Key Infrastructure (PKI) resources. Certificates generated by this program are compatible with extant industry practice, although some users might find the interpretation of X509v3 extension fields somewhat liberal. However, the identity keys are probably not compatible with anything other than Autokey.

Some files used by this program are encrypted using a private password. The **-p** option specifies the read password for local encrypted files and the **-q** option the write password for encrypted files sent to remote sites. If no password is specified, the host name returned by the Unix hostname(1) command, normally the DNS name of the host, is used as the the default read password, for convenience. The **ntp-keygen** program prompts for the password if it reads an encrypted file and the password is missing or incorrect. If an encrypted file is read successfully and no write password is specified, the read

password is used as the write password by default.

The **pw** option of the **crypto** ntpd(8) configuration command specifies the read password for previously encrypted local files. This must match the local read password used by this program. If not specified, the host name is used. Thus, if files are generated by this program without an explicit password, they can be read back by ntpd(8) without specifying an explicit password but only on the same host. If the write password used for encryption is specified as the host name, these files can be read by that host with no explicit password.

Normally, encrypted files for each host are generated by that host and used only by that host, although exceptions exist as noted later on this page. The symmetric keys file, normally called *ntp.keys*, is usually installed in */etc*. Other files and links are usually installed in */usr/local/etc*, which is normally in a shared filesystem in NFS-mounted networks and cannot be changed by shared clients. In these cases, NFS clients can specify the files in another directory such as */etc* using the **keysdir** ntpd(8) configuration file command.

This program directs commentary and error messages to the standard error stream *stderr* and remote files to the standard output stream *stdout* where they can be piped to other applications or redirected to files. The names used for generated files and links all begin with the string *ntpkey** and include the file type, generating host and filestamp, as described in the *Cryptographic Data Files* section below.

Running the Program

The safest way to run the **ntp-keygen** program is logged in directly as root. The recommended procedure is change to the *keys* directory, usually */usr/local/etc*, then run the program.

To test and gain experience with Autokey concepts, log in as root and change to the *keys* directory, usually */usr/local/etc*. When run for the first time, or if all files with names beginning with *ntpkey** have been removed, use the **ntp-keygen** command without arguments to generate a default **RSA** host key and matching **RSA-MD5** certificate file with expiration date one year hence, which is all that is necessary in many cases. The program also generates soft links from the generic names to the respective files. If run again without options, the program uses the existing keys and parameters and generates a new certificate file with new expiration date one year hence, and soft link.

The host key is used to encrypt the cookie when required and so must be **RSA** type. By default, the host key is also the sign key used to encrypt signatures. When necessary, a different sign key can be specified and this can be either **RSA** or **DSA** type. By default, the message digest type is **MD5**, but any combination of sign key type and message digest type supported by the OpenSSL library can be specified, including those using the **AES128CMAC**, **MD2**, **MD5**, **MDC2**, **SHA**, **SHA1** and **RIPE160** message digest algorithms. However, the scheme specified in the certificate must be compatible with the sign key. Certificates using any digest algorithm are compatible with **RSA** sign keys; however, only

SHA and **SHA1** certificates are compatible with **DSA** sign keys.

Private/public key files and certificates are compatible with other OpenSSL applications and very likely other libraries as well. Certificates or certificate requests derived from them should be compatible with extant industry practice, although some users might find the interpretation of X509v3 extension fields somewhat liberal. However, the identification parameter files, although encoded as the other files, are probably not compatible with anything other than Autokey.

Running the program as other than root and using the Unix su(1) command to assume root may not work properly, since by default the OpenSSL library looks for the random seed file *.rnd* in the user home directory. However, there should be only one *.rnd*, most conveniently in the root directory, so it is convenient to define the RANDFILE environment variable used by the OpenSSL library as the path to *.rnd*.

Installing the keys as root might not work in NFS-mounted shared file systems, as NFS clients may not be able to write to the shared keys directory, even as root. In this case, NFS clients can specify the files in another directory such as */etc* using the **keysdir** ntpd(8) configuration file command. There is no need for one client to read the keys and certificates of other clients or servers, as these data are obtained automatically by the Autokey protocol.

Ordinarily, cryptographic files are generated by the host that uses them, but it is possible for a trusted agent (TA) to generate these files for other hosts; however, in such cases files should always be encrypted. The subject name and trusted name default to the hostname of the host generating the files, but can be changed by command line options. It is convenient to designate the owner name and trusted name as the subject and issuer fields, respectively, of the certificate. The owner name is also used for the host and sign key files, while the trusted name is used for the identity files.

All files are installed by default in the keys directory */usr/local/etc*, which is normally in a shared filesystem in NFS-mounted networks. The actual location of the keys directory and each file can be overridden by configuration commands, but this is not recommended. Normally, the files for each host are generated by that host and used only by that host, although exceptions exist as noted later on this page.

Normally, files containing private values, including the host key, sign key and identification parameters, are permitted root read/write-only; while others containing public values are permitted world readable. Alternatively, files containing private values can be encrypted and these files permitted world readable, which simplifies maintenance in shared file systems. Since uniqueness is insured by the *hostname* and *filestamp* file name extensions, the files for an NTP server and dependent clients can all be installed in the same shared directory.

The recommended practice is to keep the file name extensions when installing a file and to install a soft link from the generic names specified elsewhere on this page to the generated files. This allows new file generations to be activated simply by changing the link. If a link is present, ntpd(8) follows it to the file name to extract the *filestamp*. If a link is not present, ntpd(8) extracts the *filestamp* from the file itself. This allows clients to verify that the file and generation times are always current. The **ntp-keygen** program uses the same *filestamp* extension for all files generated at one time, so each generation is distinct and can be readily recognized in monitoring data.

Run the command on as many hosts as necessary. Designate one of them as the trusted host (TH) using **ntp-keygen** with the **-T** option and configure it to synchronize from reliable Internet servers. Then configure the other hosts to synchronize to the TH directly or indirectly. A certificate trail is created when Autokey asks the immediately ascendant host towards the TH to sign its certificate, which is then provided to the immediately descendant host on request. All group hosts should have acyclic certificate trails ending on the TH.

The host key is used to encrypt the cookie when required and so must be RSA type. By default, the host key is also the sign key used to encrypt signatures. A different sign key can be assigned using the **-S** option and this can be either **RSA** or **DSA** type. By default, the signature message digest type is **MD5**, but any combination of sign key type and message digest type supported by the OpenSSL library can be specified using the **-c** option.

The rules say cryptographic media should be generated with proventic filestamps, which means the host should already be synchronized before this program is run. This of course creates a chicken-and-egg problem when the host is started for the first time. Accordingly, the host time should be set by some other means, such as eyeball-and-wristwatch, at least so that the certificate lifetime is within the current year. After that and when the host is synchronized to a proventic source, the certificate should be re-generated.

Additional information on trusted groups and identity schemes is on the "Autokey Public-Key Authentication" page.

File names begin with the prefix *ntpkey_* and end with the suffix *_hostname*. *filestamp*, where *hostname* is the owner name, usually the string returned by the Unix hostname(1) command, and *filestamp* is the NTP seconds when the file was generated, in decimal digits. This both guarantees uniqueness and simplifies maintenance procedures, since all files can be quickly removed by a **rm** *ntpkey** command or all files generated at a specific time can be removed by a **rm** **filestamp* command. To further reduce the risk of misconfiguration, the first two lines of a file contain the file name and generation date and time as comments.

Trusted Hosts and Groups

Each cryptographic configuration involves selection of a signature scheme and identification scheme, called a cryptotype, as explained in the *Authentication Options* section of ntp.conf(5). The default cryptotype uses **RSA** encryption, **MD5** message digest and **TC** identification. First, configure a NTP subnet including one or more low-stratum trusted hosts from which all other hosts derive synchronization directly or indirectly. Trusted hosts have trusted certificates; all other hosts have nontrusted certificates. These hosts will automatically and dynamically build authoritative certificate trails to one or more trusted hosts. A trusted group is the set of all hosts that have, directly or indirectly, a certificate trail ending at a trusted host. The trail is defined by static configuration file entries or dynamic means described on the *Automatic NTP Configuration Options* section of ntp.conf(5).

On each trusted host as root, change to the keys directory. To insure a fresh fileset, remove all *ntpkey* files. Then run **ntp-keygen -T** to generate keys and a trusted certificate. On all other hosts do the same, but leave off the **-T** flag to generate keys and nontrusted certificates. When complete, start the NTP daemons beginning at the lowest stratum and working up the tree. It may take some time for Autokey to instantiate the certificate trails throughout the subnet, but setting up the environment is completely automatic.

If it is necessary to use a different sign key or different digest/signature scheme than the default, run **ntp-keygen** with the **-S** *type* option, where *type* is either **RSA** or **DSA**. The most frequent need to do this is when a **DSA**-signed certificate is used. If it is necessary to use a different certificate scheme than the default, run **ntp-keygen** with the **-c** *scheme* option and selected *scheme* as needed. If **ntp-keygen** is run again without these options, it generates a new certificate using the same scheme and sign key, and soft link.

After setting up the environment it is advisable to update certificates from time to time, if only to extend the validity interval. Simply run **ntp-keygen** with the same flags as before to generate new certificates using existing keys, and soft links. However, if the host or sign key is changed, ntpd(8) should be restarted. When ntpd(8) is restarted, it loads any new files and restarts the protocol. Other dependent hosts will continue as usual until signatures are refreshed, at which time the protocol is restarted.

Identity Schemes

As mentioned on the Autonomous Authentication page, the default **TC** identity scheme is vulnerable to a middleman attack. However, there are more secure identity schemes available, including **PC**, **IFF**, **GQ** and **MV** schemes described below. These schemes are based on a TA, one or more trusted hosts and some number of nontrusted hosts. Trusted hosts prove identity using values provided by the TA, while the remaining hosts prove identity using values provided by a trusted host and certificate trails that end on that host. The name of a trusted host is also the name of its sugroup and also the subject and issuer name on its trusted certificate. The TA is not necessarily a trusted host in this sense, but often is.

In some schemes there are separate keys for servers and clients. A server can also be a client of another

server, but a client can never be a server for another client. In general, trusted hosts and nontrusted hosts that operate as both server and client have parameter files that contain both server and client keys. Hosts that operate only as clients have key files that contain only client keys.

The PC scheme supports only one trusted host in the group. On trusted host alice run **ntp-keygen -P -p** *password* to generate the host key file *ntpkey_* **RSA** *key_alice. filestamp* and trusted private certificate file *ntpkey_* **RSA-MD5** *_ cert_alice. filestamp*, and soft links. Copy both files to all group hosts; they replace the files which would be generated in other schemes. On each host *bob* install a soft link from the generic name *ntpkey_host_bob* to the host key file and soft link *ntpkey_cert_bob* to the private certificate file. Note the generic links are on bob, but point to files generated by trusted host alice. In this scheme it is not possible to refresh either the keys or certificates without copying them to all other hosts in the group, and recreating the soft links.

For the **IFF** scheme proceed as in the **TC** scheme to generate keys and certificates for all group hosts, then for every trusted host in the group, generate the **IFF** parameter file. On trusted host alice run **ntp-keygen -T -I -p** *password* to produce her parameter file *ntpkey_IFFpar_alice.filestamp*, which includes both server and client keys. Copy this file to all group hosts that operate as both servers and clients and install a soft link from the generic *ntpkey_iff_alice* to this file. If there are no hosts restricted to operate only as clients, there is nothing further to do. As the **IFF** scheme is independent of keys and certificates, these files can be refreshed as needed.

If a rogue client has the parameter file, it could masquerade as a legitimate server and present a middleman threat. To eliminate this threat, the client keys can be extracted from the parameter file and distributed to all restricted clients. After generating the parameter file, on alice run **ntp-keygen -e** and pipe the output to a file or email program. Copy or email this file to all restricted clients. On these clients install a soft link from the generic *ntpkey_iff_alice* to this file. To further protect the integrity of the keys, each file can be encrypted with a secret password.

For the **GQ** scheme proceed as in the **TC** scheme to generate keys and certificates for all group hosts, then for every trusted host in the group, generate the **IFF** parameter file. On trusted host alice run **ntp-keygen -T -G -p** *password* to produce her parameter file *ntpkey_GQpar_alice.filestamp*, which includes both server and client keys. Copy this file to all group hosts and install a soft link from the generic *ntpkey_gq_alice* to this file. In addition, on each host *bob* install a soft link from generic *ntpkey_gq_bob* to this file. As the **GQ** scheme updates the **GQ** parameters file and certificate at the same time, keys and certificates can be regenerated as needed.

For the **MV** scheme, proceed as in the **TC** scheme to generate keys and certificates for all group hosts. For illustration assume trish is the TA, alice one of several trusted hosts and bob one of her clients. On TA trish run **ntp-keygen -V** *n* **-p** *password*, where *n* is the number of revokable keys (typically 5) to produce the parameter file *ntpkeys_MVpar_trish.filestamp* and client key files *ntpkeys_MVkeyd _*

trish. filestamp where *d* is the key number ($0 < d < n$). Copy the parameter file to alice and install a soft link from the generic *ntpkey_mv_alice* to this file. Copy one of the client key files to alice for later distribution to her clients. It does not matter which client key file goes to alice, since they all work the same way. Alice copies the client key file to all of her clients. On client bob install a soft link from generic *ntpkey_mvkey_bob* to the client key file. As the **MV** scheme is independent of keys and certificates, these files can be refreshed as needed.

Command Line Options

-b --imbits= *modulus*

Set the number of bits in the identity modulus for generating identity keys to *modulus* bits. The number of bits in the identity modulus defaults to 256, but can be set to values from 256 to 2048 (32 to 256 octets). Use the larger moduli with caution, as this can consume considerable computing resources and increases the size of authenticated packets.

-c --certificate= *scheme*

Select certificate signature encryption/message digest scheme. The *scheme* can be one of the following: **RSA-MD2**, **RSA-MD5**, **RSA-MDC2**, **RSA-SHA**, **RSA-SHA1**, **RSA-RIPEMD160**, **DSA-SHA**, or **DSA-SHA1**. Note that **RSA** schemes must be used with an **RSA** sign key and **DSA** schemes must be used with a **DSA** sign key. The default without this option is **RSA-MD5**. If compatibility with FIPS 140-2 is required, either the **DSA-SHA** or **DSA-SHA1** scheme must be used.

-C --cipher= *cipher*

Select the OpenSSL cipher to encrypt the files containing private keys. The default without this option is three-key triple DES in CBC mode, **des-ede3-cbc**. The **openssl -h** command provided with OpenSSL displays available ciphers.

-d --debug-level

Increase debugging verbosity level. This option displays the cryptographic data produced in eye-friendly billboards.

-D --set-debug-level= *level*

Set the debugging verbosity to *level*. This option displays the cryptographic data produced in eye-friendly billboards.

-e --id-key

Write the **IFF** or **GQ** public parameters from the *IFFkey or GQkey* client keys file previously specified as unencrypted data to the standard output stream *stdout*. This is intended for automatic key distribution by email.

-G --gq-params

Generate a new encrypted **GQ** parameters and key file for the Guillou-Quisquater (GQ) identity scheme. This option is mutually exclusive with the **-I** and **-V** options.

-H --host-key

Generate a new encrypted **RSA** public/private host key file.

-I --iffkey

Generate a new encrypted **IFF** key file for the Schnorr (IFF) identity scheme. This option is mutually exclusive with the **-G** and Fl V options.

-i --ident= *group*

Set the optional Autokey group name to *group*. This is used in the identity scheme parameter file names of **IFF**, **GQ**, and **MV** client parameters files. In that role, the default is the host name if no group is provided. The group name, if specified using **-i** or **-s** following an '@' character, is also used in certificate subject and issuer names in the form *host @ group* and should match the group specified via **crypto ident** or **server ident** in the ntpd configuration file.

-l --lifetime= *days*

Set the lifetime for certificate expiration to *days*. The default lifetime is one year (365 days).

-m --modulus= *bits*

Set the number of bits in the prime modulus for generating files to *bits*. The modulus defaults to 512, but can be set from 256 to 2048 (32 to 256 octets). Use the larger moduli with caution, as this can consume considerable computing resources and increases the size of authenticated packets.

-M --md5key

Generate a new symmetric keys file containing 10 **MD5** keys, and if OpenSSL is available, 10 **SHA** keys. An **MD5** key is a string of 20 random printable ASCII characters, while a **SHA** key is a string of 40 random hex digits. The file can be edited using a text editor to change the key type or key content. This option is mutually exclusive with all other options.

-p --password= *passwd*

Set the password for reading and writing encrypted files to *passwd*. These include the host, sign and identify key files. By default, the password is the string returned by the Unix **hostname** command.

-P --pvt-cert

Generate a new private certificate used by the **PC** identity scheme. By default, the program

generates public certificates. Note: the PC identity scheme is not recommended for new installations.

-q --export-passwd= *passwd*

Set the password for writing encrypted **IFF**, **GQ and MV** identity files redirected to *stdout* to *passwd*. In effect, these files are decrypted with the **-p** password, then encrypted with the **-q** password. By default, the password is the string returned by the Unix **hostname** command.

-s --subject-key= *file* ... [host] [*@ group*]

Specify the Autokey host name, where *host* is the optional host name and *group* is the optional group name. The host name, and if provided, group name are used in *host @ group* form as certificate subject and issuer. Specifying **-s -@** *group* is allowed, and results in leaving the host name unchanged, as with **-i** *group*. The group name, or if no group is provided, the host name are also used in the file names of **IFF**, **GQ**, and **MV** identity scheme client parameter files. If *host* is not specified, the default host name is the string returned by the Unix **hostname** command.

-S --sign-key= [**RSA** | **DSA**]

Generate a new encrypted public/private sign key file of the specified type. By default, the sign key is the host key and has the same type. If compatibility with FIPS 140-2 is required, the sign key type must be **DSA**.

-T --trusted-cert

Generate a trusted certificate. By default, the program generates a non-trusted certificate.

-V --mv-params *nkeys*

Generate *nkeys* encrypted server keys and parameters for the Mu-Varadharajan (MV) identity scheme. This option is mutually exclusive with the **-I** and **-G** options. Note: support for this option should be considered a work in progress.

Random Seed File

All cryptographically sound key generation schemes must have means to randomize the entropy seed used to initialize the internal pseudo-random number generator used by the library routines. The OpenSSL library uses a designated random seed file for this purpose. The file must be available when starting the NTP daemon and **ntp-keygen** program. If a site supports OpenSSL or its companion OpenSSH, it is very likely that means to do this are already available.

It is important to understand that entropy must be evolved for each generation, for otherwise the random number sequence would be predictable. Various means dependent on external events, such as keystroke intervals, can be used to do this and some systems have built-in entropy sources. Suitable means are

described in the OpenSSL software documentation, but are outside the scope of this page.

The entropy seed used by the OpenSSL library is contained in a file, usually called *.rnd*, which must be available when starting the NTP daemon or the **ntp-keygen** program. The NTP daemon will first look for the file using the path specified by the **randfile** subcommand of the **crypto** configuration command. If not specified in this way, or when starting the **ntp-keygen** program, the OpenSSL library will look for the file using the path specified by the RANDFILE environment variable in the user home directory, whether root or some other user. If the RANDFILE environment variable is not present, the library will look for the *.rnd* file in the user home directory. Since both the **ntp-keygen** program and ntpd(8) daemon must run as root, the logical place to put this file is in */.rnd* or */root/.rnd*. If the file is not available or cannot be written, the daemon exits with a message to the system log and the program exits with a suitable error message.

Cryptographic Data Files

All file formats begin with two nonencrypted lines. The first line contains the file name, including the generated host name and filestamp, in the format *ntpkey_key _ name*. *filestamp*, where *key* is the key or parameter type, *name* is the host or group name and *filestamp* is the filestamp (NTP seconds) when the file was created. By convention, *key* names in generated file names include both upper and lower case characters, while *key* names in generated link names include only lower case characters. The filestamp is not used in generated link names. The second line contains the datestamp in conventional Unix *date* format. Lines beginning with '#' are considered comments and ignored by the **ntp-keygen** program and ntpd(8) daemon.

The remainder of the file contains cryptographic data, encoded first using ASN.1 rules, then encrypted if necessary, and finally written in PEM-encoded printable ASCII text, preceded and followed by MIME content identifier lines.

The format of the symmetric keys file, ordinarily named *ntp.keys*, is somewhat different than the other files in the interest of backward compatibility. Ordinarily, the file is generated by this program, but it can be constructed and edited using an ordinary text editor.

```
# ntpkey_MD5key_bk.ntp.org.3595864945
# Thu Dec 12 19:22:25 2013
1  MD5 L";Nw<'.I<f4U0)247"i  # MD5 key
2  MD5 &>l0%XXK9O'51VwV<xq~  # MD5 key
3  MD5 lb4zLW~d^!K:]RsD'qb6  # MD5 key
4  MD5 Yue:tL[+vR)M'n~bY,'?  # MD5 key
5  MD5 B;fx'Kgr/&4ZTbL6=RxA  # MD5 key
6  MD5 4eYwa'o}3i@@V@..R9!l  # MD5 key
7  MD5 'A.([h+;wTQ|xfi%Sn_!  # MD5 key
```

```
8  MD5 45:V,r4]l6y^JH6"Sh?F  # MD5 key
9  MD5 3-5vcn*6l29DS?Xdsg)*  # MD5 key
10 MD5 2late4Me            # MD5 key
11 SHA1 a27872d3030a9025b8446c751b4551a7629af65c  # SHA1 key
12 SHA1 21bc3b4865dbb9e920902abdccb3e04ff97a5e74  # SHA1 key
13 SHA1 2b7736fe24fef5ba85ae11594132ab5d6f6daba9  # SHA1 key
14 SHA  a5332809c8878dd3a5b918819108a111509aeceb  # SHA  key
15 MD2  2fe16c88c760ff2f16d4267e36c1aa6c926e6964  # MD2  key
16 MD4  b2691811dc19cfc0e2f9bcacd74213f29812183d  # MD4  key
17 MD5  e4d6735b8bdad58ec5ffcb087300a17f7fef1f7c  # MD5  key
18 MDC2 a8d5e2315c025bf3a79174c87fbd10477de2eabc  # MDC2 key
19 RIPEMD160 77ca332cafb30e3cafb174dcd5b80ded7ba9b3d2  # RIPEMD160 key
20 AES128CMAC f92ff73eee86c1e7dc638d6489a04e4e555af878  # AES128CMAC key
```

Figure 1. Typical Symmetric Key File

Figure 1 shows a typical symmetric keys file used by the reference implementation. Following the header the keys are entered one per line in the format

keyno type key

where *keyno* is a positive integer in the range 1-65535; *type* is the key type for the message digest algorithm, which in the absence of the OpenSSL library must be **MD5** to designate the MD5 message digest algorithm; if the OpenSSL library is installed, the key type can be any message digest algorithm supported by that library; however, if compatibility with FIPS 140-2 is required, the key type must be either **SHA** or **SHA1**; *key* is the key itself, which is a printable ASCII string 20 characters or less in length: each character is chosen from the 93 printable characters in the range 0x21 through 0x7e ('!' through '~') excluding space and the '#' character, and terminated by whitespace or a '#' character. An OpenSSL key consists of a hex-encoded ASCII string of 40 characters, which is truncated as necessary.

Note that the keys used by the ntpq(8) and ntpdc(8) programs are checked against passwords requested by the programs and entered by hand, so it is generally appropriate to specify these keys in human readable ASCII format.

The **ntp-keygen** program generates a symmetric keys file *ntpkey_MD5key_hostname.filestamp*. Since the file contains private shared keys, it should be visible only to root and distributed by secure means to other subnet hosts. The NTP daemon loads the file *ntp.keys*, so **ntp-keygen** installs a soft link from this name to the generated file. Subsequently, similar soft links must be installed by manual or automated means on the other subnet hosts. While this file is not used with the Autokey Version 2 protocol, it is needed to authenticate some remote configuration commands used by the ntpq(8) and ntpdc(8) utilities.

OPTIONS

-b *imbits*, **--imbits**=*imbits*

identity modulus bits. This option takes an integer number as its argument. The value of *imbits* is constrained to being:
in the range 256 through 2048

The number of bits in the identity modulus. The default is 256.

-c *scheme*, **--certificate**=*scheme*
certificate scheme.

scheme is one of RSA-MD2, RSA-MD5, RSA-MDC2, RSA-SHA, RSA-SHA1, RSA-RIPEMD160, DSA-SHA, or DSA-SHA1.

Select the certificate signature encryption/message digest scheme. Note that RSA schemes must be used with a RSA sign key and DSA schemes must be used with a DSA sign key. The default without this option is RSA-MD5.

-C *cipher*, **--cipher**=*cipher*
privatekey cipher.

Select the cipher which is used to encrypt the files containing private keys. The default is three-key triple DES in CBC mode, equivalent to "**-C des-ede3-cbc**". The openssl tool lists ciphers available in "**openssl -h**" output.

-d, **--debug-level**
Increase debug verbosity level. This option may appear an unlimited number of times.

-D *number*, **--set-debug-level**=*number*
Set the debug verbosity level. This option may appear an unlimited number of times. This option takes an integer number as its argument.

-e, **--id-key**
Write IFF or GQ identity keys.

Write the public parameters from the IFF or GQ client keys to the standard output. This is intended for automatic key distribution by email.

-G, **--gq-params**
Generate GQ parameters and keys.

Generate parameters and keys for the GQ identification scheme, obsoleting any that may exist.

-H, **--host-key**
generate RSA host key.

Generate new host keys, obsoleting any that may exist.

-I, **--iffkey**
generate IFF parameters.

Generate parameters for the IFF identification scheme, obsoleting any that may exist.

-i *group*, **--ident**=*group*
set Autokey group name.

Set the optional Autokey group name to name. This is used in the file name of IFF, GQ, and MV client parameters files. In that role, the default is the host name if this option is not provided. The group name, if specified using **-i/--ident** or using **-s/--subject-name** following an '**@**' character, is also a part of the self-signed host certificate subject and issuer names in the form **host@group** and should match the '**crypto ident**' or '**server ident**' configuration in the **ntpd** configuration file.

-l *lifetime*, **--lifetime**=*lifetime*
set certificate lifetime. This option takes an integer number as its argument.

Set the certificate expiration to lifetime days from now.

-m *modulus*, **--modulus**=*modulus*
prime modulus. This option takes an integer number as its argument. The value of *modulus* is constrained to being:
in the range 256 through 2048

The number of bits in the prime modulus. The default is 512.

-M, **--md5key**
generate symmetric keys.

Generate symmetric keys, obsoleting any that may exist.

-P, **--pvt-cert**
generate PC private certificate.

Generate a private certificate. By default, the program generates public certificates.

-p *passwd*, **--password**=*passwd*
local private password.

Local files containing private data are encrypted with the DES-CBC algorithm and the specified password. The same password must be specified to the local ntpd via the "crypto pw password" configuration command. The default password is the local hostname.

-q *passwd*, **--export-passwd**=*passwd*
export IFF or GQ group keys with password.

Export IFF or GQ identity group keys to the standard output, encrypted with the DES-CBC algorithm and the specified password. The same password must be specified to the remote ntpd via the "crypto pw password" configuration command. See also the option --id-key (-e) for unencrypted exports.

-s *host@group*, **--subject-name**=*host@group*
set host and optionally group name.

Set the Autokey host name, and optionally, group name specified following an '**@**' character. The host name is used in the file name of generated host and signing certificates, without the group name. The host name, and if provided, group name are used in **host@group** form for the host certificate subject and issuer fields. Specifying '**-s @group**' is allowed, and results in leaving the host name unchanged while appending **@group** to the subject and issuer fields, as with **-i group**. The group name, or if not provided, the host name are also used in the file names of IFF, GQ, and MV client parameter files.

-S *sign*, **--sign-key**=*sign*
generate sign key (RSA or DSA).

Generate a new sign key of the designated type, obsoleting any that may exist. By default, the program uses the host key as the sign key.

-T, **--trusted-cert**
trusted certificate (TC scheme).

Generate a trusted certificate. By default, the program generates a non-trusted certificate.

-V *num*, **--mv-params**=*num*

generate <num> MV parameters. This option takes an integer number as its argument.

Generate parameters and keys for the Mu-Varadharajan (MV) identification scheme.

-v *num*, **--mv-keys**=*num*
update <num> MV keys. This option takes an integer number as its argument.

This option has not been fully documented.

-?, **--help**
Display usage information and exit.

-!, **--more-help**
Pass the extended usage information through a pager.

-> [*cfgfile*], **--save-opts** [=*cfgfile*]
Save the option state to *cfgfile*. The default is the *last* configuration file listed in the **OPTION PRESETS** section, below. The command will exit after updating the config file.

-< *cfgfile*, **--load-opts**=*cfgfile*, **--no-load-opts**
Load options from *cfgfile*. The *no-load-opts* form will disable the loading of earlier config/rc/ini files. *--no-load-opts* is handled early, out of order.

--version [{*v*|*c*|*n*}]
Output version of program and exit. The default mode is 'v', a simple version. The 'c' mode will print copyright information and 'n' will print the full copyright notice.

OPTION PRESETS

Any option that is not marked as *not presettable* may be preset by loading values from configuration ("RC" or ".INI") file(s) and values from environment variables named:
NTP_KEYGEN_<option-name> or **NTP_KEYGEN**
The environmental presets take precedence (are processed later than) the configuration files. The *homerc* files are "*$HOME*", and ".". If any of these are directories, then the file *.ntprc* is searched for within those directories.

USAGE

ENVIRONMENT

See **OPTION PRESETS** for configuration environment variables.

FILES

See **OPTION PRESETS** for configuration files.

EXIT STATUS

One of the following exit values will be returned:

0 (EXIT_SUCCESS)
: Successful program execution.

1 (EXIT_FAILURE)
: The operation failed or the command syntax was not valid.

66 (EX_NOINPUT)
: A specified configuration file could not be loaded.

70 (EX_SOFTWARE)
: libopts had an internal operational error. Please report it to autogen-users@lists.sourceforge.net. Thank you.

AUTHORS

The University of Delaware and Network Time Foundation

COPYRIGHT

BUGS

It can take quite a while to generate some cryptographic values.

Please report bugs to http://bugs.ntp.org .

Please send bug reports to: http://bugs.ntp.org, bugs@ntp.org

NOTES

Portions of this document came from FreeBSD.

This manual page was *AutoGen*-erated from the **ntp-keygen** option definitions.