NAME

ntp_auth - Authentication Options

INTRODUCTION

This page describes the various cryptographic authentication provisions in NTPv4. Details about the configuration commands and options are given on the Configuration Options page. Details about the automatic server discovery schemes are described on the Automatic Server Discovery Schemes page. Additional information is available in the papers, reports, memoranda and briefings cited on the NTP Project page. Authentication support allows the NTP client to verify that servers are in fact known and trusted and not intruders intending accidentally or intentionally to masquerade as a legitimate server.

The NTPv3 specification RFC-1305 defines a scheme properly described as symmetric key cryptography. It uses the Data Encryption Standard (DES) algorithm operating in cipher-block chaining (CBC) mode. Subsequently, this scheme was replaced by the RSA Message Digest 5 (MD5) algorithm commonly called keyed-MD5. Either algorithm computes a message digest or one-way hash which can be used to verify the client has the same key and key identifier as the server. If the OpenSSL cryptographic library is installed, support is available for all algorithms included in the library. Note however, if conformance to FIPS 140-2 is required, only a limited subset of these algorithms is available.

NTPv4 includes the NTPv3 scheme and optionally a new scheme based on public key cryptography and called Autokey. Public key cryptography is generally considered more secure than symmetric key cryptography, since the security is based on private and public values which are generated by each participant and where the private value is never revealed. Autokey uses X.509 public certificates, which can be produced by commercial services, utility programs in the OpenSSL software library or the **ntp-keygen** utility program in the NTP software distribution.

While the algorithms for MD5 symmetric key cryptography are included in the NTPv4 software distribution, modern algorithms for symmetric key and public key cryptograpny requires the OpenSSL software library to be installed before building the NTP distribution. This library is available from http://www.openssl.org and can be installed using the procedures outlined in the Building and Installing the Distribution page. Once installed, the configure and build process automatically detects the library and links the library routines required.

Note that according to US law, NTP binaries including OpenSSL library components, including the OpenSSL library itself, cannot be exported outside the US without license from the US Department of Commerce. Builders outside the US are advised to obtain the OpenSSL library directly from OpenSSL, which is outside the US, and build outside the US.

Authentication is configured separately for each association using the **key** or **autokey** option of the **server** configuration command, as described in the Server Options page, and the options described on this page. The ntp-keygen page describes the files required for the various authentication schemes. Further details are in the briefings, papers and reports at the NTP project page linked from www.ntp.org.

SYMMETRIC KEY CRYPTOGRAPHY

The original RFC-1305 specification allows any one of possibly 65,534 keys (excluding zero), each distinguished by a 32-bit key ID, to authenticate an association. The servers and clients involved must agree on the key, key ID and key type to authenticate NTP packets. If an NTP packet includes a message authentication code (MAC), consisting of a key ID and message digest, it is accepted only if the key ID matches a trusted key and the message digest is verified with this key. Note that for historic reasons the message digest algorithm is not consistent with RFC-1828. The digest is computed directly from the concatenation of the key string followed by the packet contents with the exception of the MAC itself.

Keys and related information are specified in a keys file, usually called **ntp.keys**, which must be distributed and stored using secure means beyond the scope of the NTP protocol itself. Besides the keys used for

ordinary NTP associations, additional keys can be used as passwords for the **ntpq** and **ntpdc** utility programs. Ordinarily, the **ntp.keys** file is generated by the **ntp-keygen** program, but it can be constructed and edited using an ordinary text editor. The program generates pseudo-random keys, one key for each line. Each line consists of three fields, the key identifier as a decimal number from 1 to 65534 inclusive, a key type chosen from the keywords of the **digest** option of the **crypto** command, and a 20-character printable ASCII string or a 40-character hex string as the key itself.

When **ntpd** is first started, it reads the key file specified by the **keys** command and installs the keys in the key cache. However, individual keys must be activated with the **trustedkey** configuration command before use. This allows, for instance, the installation of possibly several batches of keys and then activating a key remotely using **ntpdc**. The **requestkey** command selects the key ID used as the password for the **ntpdc** utility, while the **controlkey** command selects the key ID used as the password for the **ntpq** utility.

By default, the message digest algorithm is MD5 selected by the key type **M** in the keys file. However, if the OpenSSL library is installed, any message digest algorithm supported by that library can be used. The key type is selected as the algorithm name given in the OpenSSL documentation. The key type is associated with the key and can be different for different keys. The server and client must share the same key, key ID and key type and both must be trusted. Note that if conformance to FIPS 140-2 is required, the message digest algorithm must conform to the Secure Hash Standard (SHS), which requires an algorithm from the Secure Hash Algorithm (SHA) family, and the digital signature encryption algorithm, if used, must conform to the Digital Signature Standard (DSS), which requires the Digital Signature Algorithm (DSA).

In addition to the above means, **ntpd** now supports Microsoft Windows MS-SNTP authentication using Active Directory services. This support was contributed by the Samba Team and is still in development. It is enabled using the **mssntp** flag of the **restrict** command described on the Access Control Options page. Note: Potential users should be aware that these services involve a TCP connection to another process that could potentially block, denying services to other users. Therefore, this flag should be used only for a dedicated server with no clients other than MS-SNTP.

PUBLIC KEY CRYPTOGRAPHY

NTPv4 supports the Autokey security protocol, which is based on public key cryptography. The Autokey Version 2 protocol described on the Autokey Protocol page verifies packet integrity using MD5 message digests and verifies the source using digital signatures and any of several digest/signature schemes. Optional identity schemes described on the Autokey Identity Schemes page are based on cryptographic challenge/response exchanges. These schemes provide strong security against replay with or without message modification, spoofing, masquerade and most forms of clogging attacks. These schemes are described along with an executive summary, current status, briefing slides and reading list on the Autonomous Authentication page.

Autokey authenticates individual packets using cookies bound to the IP source and destination addresses. The cookies must have the same addresses at both the server and client. For this reason operation with network address translation schemes is not possible. This reflects the intended robust security model where government and corporate NTP servers are operated outside firewall perimeters.

There are three timeouts associated with the Autokey scheme. The key list timeout, which defaults to about 1.1 h, specifies the interval between generating new key lists. The revoke timeout, which defaults to about 36 h, specifies the interval between generating new private values. The restart timeout, with default about 5 d, specifies the interval between protocol restarts to refresh public values. In general, the behavior when these timeouts expire is not affected by the issues discussed on this page.

NTP SECURE GROUPS

NTP secure groups are used to define cryptographic compartments and security hierarchies. All hosts belonging to a secure group have the same group name but different host names. The string specified in the **host** option of the **crypto** command is the name of the host and the name used in the host key, sign key and

certificate files. The string specified in the **ident** option of the **crypto** command is the group name of all group hosts and the name used in the identity files. The file naming conventions are described on the ntp-keygen page.

Each group includes one or more trusted hosts (THs) operating at the root, or lowest stratum in the group. The group name is used in the subject and issuer fields of the TH self-signed trusted certificate for these hosts. The host name is used in the subject and issuer fields of the self-signed certificates for all other hosts.

All group hosts are configured to provide an unbroken path, called a certificate trail, from each host, possibly via intermediate hosts and ending at a TH. When a host starts up, it recursively retrieves the certificates along the trail in order to verify group membership and avoid masquerade and middleman attacks.

Secure groups can be configured as hierarchies where a TH of one group can be a client of one or more other groups operating at a lower stratum. A certificate trail consist of a chain of hosts starting at a client, leading through secondary servers of progressively lower stratum and ending at a TH. In one scenario, groups RED and GREEN can be cryptographically distinct, but both be clients of group BLUE operating at a lower stratum. In another scenario, group CYAN can be a client of multiple groups YELLOW and MAGENTA, both operating at a lower stratum. There are many other scenarios, but all must be configured to include only acyclic certificate trails.

IDENTITY SCHEMES AND CRYPTOTYPES

All configurations include a public/private host key pair and matching certificate. Absent an identity scheme, this is a Trusted Certificate (TC) scheme. There are three identity schemes, IFF, GQ and MV described on the Identity Schemes page. With these schemes all servers in the group have encrypted server identity keys, while clients have nonencrypted client identity parameters. The client parameters can be obtained from a trusted agent (TA), usually one of the THs of the lower stratum group. Further information on identity schemes is on the Autokey Identity Schemes page.

A specific combination of authentication and identity schemes is called a cryptotype, which applies to clients and servers separately. A group can be configured using more than one cryptotype combination, although not all combinations are interoperable. Note however that some cryptotype combinations may successfully intemperate with each other, but may not represent good security practice. The server and client cryptotypes are defined by the the following codes.

NONE A client or server is type NONE if authentication is not available or not configured. Packets exchanged between client and server have no MAC.

AUTH A client or server is type AUTH if the **key** option is specified with the **server** configuration command and the client and server keys are compatible. Packets exchanged between clients and servers have a MAC.

PC A client or server is type PC if the **autokey** option is specified with the **server** configuration command and compatible host key and private certificate files are present. Packets exchanged between clients and servers have a MAC.

TC A client or server is type TC if the **autokey** option is specified with the **server** configuration command and compatible host key and public certificate files are present. Packets exchanged between clients and servers have a MAC.

IDENT A client or server is type IDENT if the **autokey** option is specified with the **server** configuration command and compatible host key, public certificate and identity scheme files are present. Packets exchanged between clients and servers have a MAC.

The compatible cryptotypes for clients and servers are listed in the following table.

Client/Server	NONE	AUTH	PC	TC	IDENT
NONE	yes	yes*	yes*	yes*	yes*
AUTH	no	yes	no	no	no
PC	no	no	yes	no	no
TC	no	no	no	yes	yes
IDENT	no	no	no	no	yes

* These combinations are not valid if the restriction list includes the **notrust** option.

CONFIGURATION

Autokey has an intimidating number of configuration options, most of which are not necessary in typical scenarios. The simplest scenario consists of a TH where the host name of the TH is also the name of the group. For the simplest identity scheme TC, the TH generates host key and trusted certificate files using the **ntp-keygen -T** command, while the remaining group hosts use the same command with no options to generate the host key and public certificate files. All hosts use the **crypto** configuration command with no options. Configuration with passwords is described in the ntp-keygen page. All group hosts are configured as an acyclic tree with root the TH.

When an identity scheme is included, for example IFF, the TH generates host key, trusted certificate and private server identity key files using the **ntp-keygen -T -I -i** *group* command, where *group* is the group name. The remaining group hosts use the same command as above. All hosts use the **crypto ident group** configuration command.

Hosts with no dependent clients can retrieve client parameter files from an archive or web page. The **ntp-keygen** can export these data using the **-e** option. Hosts with dependent clients other than the TH must retrieve copies of the server key files using secure means. The **ntp-keygen** can export these data using the **-q** option. In either case the data are installed as a file and then renamed using the name given as the first line in the file, but without the filestamp.

EXAMPLES

Consider a scenario involving three secure groups RED, GREEN and BLUE. RED and BLUE are typical of national laboratories providing certified time to the Internet at large. As shown ion the figure, RED TH mort and BLUE TH macabre run NTP symmetric mode with each other for monitoring or backup. For the purpose of illustration, assume both THs are primary servers. GREEN is typical of a large university providing certified time to the campus community. GREEN TH howland is a broadcast client of both RED and BLUE. BLUE uses the IFF scheme, while both RED and GREEN use the GQ scheme, but with different keys. YELLOW is a client of GREEN and for purposes of illustration a TH for YELLOW.

The BLUE TH macabre uses configuration commands

crypto pw qqsv ident blue peer mort autokey broadcast *address* **autokey**

where **qqsv** is the password for macabre files and *address* is the broadcast address for the local LAN. It generates BLUE files using the commands

ntp-keygen -p qqsv -T -G -i blue ntp-keygen -p qqsv -e >ntpkey_gqpar_blue

The first line generates the host, trusted certificate and private GQ server keys file. The second generates the public GQ client parameters file, which can have any nonconflicting mnemonic name.

The RED TH mort uses configuration commands

crypto pw xxx ident red peer macabre autokey broadcast *address* **autokey**

where **xxx** is the password for mort files. It generates RED files using the commands

ntp-keygen -p xxx -T -I -i red ntp-keygen -p xxx -e >ntpkey_iffpar_red

The GREEN TH howland uses configuration commands

crypto pw yyy ident green broadcastclient

where **yyy** is the password for howland files. It generates GREEN files using the commands

ntp-keygen -p yyy -T -G -i green ntp-keygen -p yyy -e >ntpkey_gqpar_green ntp-keygen -p yyy -q zzz >zzz_ntpkey_gqkey_green

The first two lines serve the same purpose as the preceding examples. The third line generates a copy of the private GREEN server file for use on another server in the same group, say YELLOW, but encrypted with the **zzz** password.

A client of GREEN, for example YELLOW, uses the configuration commands

crypto pw abc ident green server howland autokey

where **abc** is the password for its files. It generates files using the command

ntp-keygen -p abc

The client retrieves the client file for that group from a public archive or web page using nonsecure means. In addition, each server in a group retrieves the private server keys file from the TH of that group, but it is encrypted and so must be sent using secure means. The files are installed in the keys directory with name taken from the first line in the file, but without the filestamp.

Note that if servers of different groups, in this case RED and BLUE, share the same broadcast media, each server must have client files for all groups other than its own, while each client must have client files for all groups. Note also that this scenario is for illustration only and probably would not be wise for practical use, as if one of the TH reference clocks fails, the certificate trail becomes cyclic. In such cases the symmetric path between RED and BLUE, each in a different group, would not be a good idea.

AUTHENTICATION COMMANDS

automax [*logsec***]**
: Specifies the interval between regenerations of the session key list used with the Autokey protocol, as a power of 2 in seconds. Note that the size of the key list for each association depends on this interval and the current poll interval. The default interval is 12 (about 1.1 h). For poll intervals above the specified interval, a session key list with a single entry will be regenerated for every message sent.

controlkey *keyid*
: Specifies the key ID to use with the **ntpq** utility, which uses the standard protocol defined in RFC-1305. The *keyid* argument is the key ID for a trusted key, where the value can be in the range 1 to 65534, inclusive.

crypto [randfile *file***] [host** *name***] [ident** *name***] [pw** *password***]**
: This command requires the OpenSSL library. It activates public key cryptography and loads the required host key and public certificate. If one or more files are left unspecified, the default names are used as described below. Unless the complete path and name of the file are specified, the location of a file is relative to the keys directory specified in the **keysdir** configuration command or default **/etc/ntp/crypto**. Following are the options.

digest MD2 | **MD4** | **MD5** | **MDC2** | **RIPEMD160** | **SHA** | **SHA1**
Specify the message digest algorithm, with default MD5. If the OpenSSL library is installed, *name* can be be any message digest algorithm supported by the library not exceeding 160 bits in length. However, all Autokey participants in an Autokey subnet must use the same algorithm. Note that the Autokey message digest algorithm is separate and distinct form the symmetric key message digest algorithms. Note: If compliance with FIPS 140-2 is required, the algorithm must be ether **SHA** or **SHA1**.

host *name*
Specifies the string used when constructing the names for the host, sign and certificate files generated by the **ntp-keygen** program with the **-s** *name* option.

ident *name*
Specifies the string used in constructing the identity files generated by the **ntp-keygen** program with the **-i** *name* option.

pw *password*
Specifies the password to decrypt files previously encrypted by the **ntp-keygen** program with the **-p** option.

randfile *file*
Specifies the location of the random seed file used by the OpenSSL library. The defaults are described on the **ntp-keygen** page.

keys *keyfile*
Specifies the complete path to the MD5 key file containing the keys and key IDs used by **ntpd**, **ntpq** and **ntpdc** when operating with symmetric key cryptography. This is the same operation as the **-k** command line option. Note that the directory path for Autokey media is specified by the **keysdir** command.

keysdir *path*K
This command specifies the default directory path for Autokey cryptographic keys, parameters and certificates. The default is **/etc/ntp/crypto**. Note that the path for the symmetric keys file is specified by the **keys** command.

requestkey *keyid*
Specifies the key ID to use with the **ntpdc** utility program, which uses a proprietary protocol specific to this implementation of **ntpd**. The *keyid* argument is a key ID for a trusted key, in the range 1 to 65534, inclusive.

revoke [*logsec***]**
Specifies the interval between re-randomization of certain cryptographic values used by the Autokey scheme, as a power of 2 in seconds. These values need to be updated frequently in order to deflect brute-force attacks on the algorithms; however, updating some values is a relatively expensive operation. The default interval is 17 (about 36 h). For poll intervals above the specified interval, the values will be updated for every message sent.

trustedkey [*keyid* | **(***lowid* **...** *highid***)] [...]**
Specifies the key ID(s) which are trusted for the purposes of authenticating peers with symmetric key cryptography. Key IDs used to authenticate **ntpq** and **ntpdc** operations must be listed here and additionally be enabled with controlkey and/or requestkey. The authentication procedure for time transfer require that both the local and remote NTP servers employ the same key ID and secret for this purpose, although different keys IDs may be used with different servers. Ranges of trusted key IDs may be specified: "**trustedkey (1 ... 19) 1000 (100 ... 199)**" enables the lowest 120 key IDs which start with the digit 1. The spaces surrounding the ellipsis are required when specifying a range.

ERROR CODES

Errors can occur due to mismatched configurations, unexpected protocol restarts, expired certificates and unfriendly people. In most cases the protocol state machine recovers automatically by retransmission, time-out and restart, where necessary. Some errors are due to mismatched keys, digest schemes or identity schemes and must be corrected by installing the correct media and/or correcting the configuration file. One of the most common errors is expired certificates, which must be regenerated and signed at least once per year using the **ntp-keygen** - generate public and private keys program.

The following error codes are reported via the NTP control and monitoring protocol trap mechanism and to the **cryptostats** monitoring file if configured.

101 bad field format or length
: The packet has invalid version, length or format.

102 bad timestamp
: The packet timestamp is the same or older than the most recent received. This could be due to a replay or a server clock time step.

103 bad filestamp
: The packet filestamp is the same or older than the most recent received. This could be due to a replay or a key file generation error.

104 bad or missing public key
: The public key is missing, has incorrect format or is an unsupported type.

105 unsupported digest type
: The server requires an unsupported digest/signature scheme.

106 unsupported identity type
: The client or server has requested an identity scheme the other does not support.

107 bad signature length
: The signature length does not match the current public key.

108 signature not verified
: The message fails the signature check. It could be bogus or signed by a different private key.

109 certificate not verified
: The certificate is invalid or signed with the wrong key.

110 host certificate expired
: The old server certificate has expired.

111 bad or missing cookie
: The cookie is missing, corrupted or bogus.

112 bad or missing leapseconds table
: The leapseconds table is missing, corrupted or bogus.

113 bad or missing certificate
: The certificate is missing, corrupted or bogus.

114 bad or missing group key
: The identity key is missing, corrupt or bogus.

115 protocol error
: The protocol state machine has wedged due to unexpected restart.

FILES

See the **ntp-keygen** page. Note that provisions to load leap second values from the NIST files have been removed. These provisions are now available whether or not the OpenSSL library is available. However, the functions that can download these values from servers remains available.

SEE ALSO

ntp.conf(5), ntpd(8)

The official HTML documentation.

This file was automatically generated from HTML source.