NAME

openssl−ts, ts − Time Stamping Authority tool (client/server)

SYNOPSIS

openssl ts −query [**−rand file...**] [**−writerand file**] [**−config** configfile] [**−data** file_to_hash] [**−digest** digest_bytes] [**−*digest***] [**−tspolicy** object_id] [**−no_nonce**] [**−cert**] [**−in** request.tsq] [**−out** request.tsq] [**−text**]

openssl ts −reply [**−config** configfile] [**−section** tsa_section] [**−queryfile** request.tsq] [**−passin** password_src] [**−signer** tsa_cert.pem] [**−inkey** file_or_id] [**−*digest***] [**−chain** certs_file.pem] [**−tspolicy** object_id] [**−in** response.tsr] [**−token_in**] [**−out** response.tsr] [**−token_out**] [**−text**] [**−engine** id]

openssl ts −verify [**−data** file_to_hash] [**−digest** digest_bytes] [**−queryfile** request.tsq] [**−in** response.tsr] [**−token_in**] [**−CApath** trusted_cert_path] [**−CAfile** trusted_certs.pem] [**−untrusted** cert_file.pem] [*verify options*]

verify options: [−attime timestamp] [−check_ss_sig] [−crl_check] [−crl_check_all] [−explicit_policy] [−extended_crl] [−ignore_critical] [−inhibit_any] [−inhibit_map] [−issuer_checks] [−no_alt_chains] [−no_check_time] [−partial_chain] [−policy arg] [−policy_check] [−policy_print] [−purpose purpose] [−suiteB_128] [−suiteB_128_only] [−suiteB_192] [−trusted_first] [−use_deltas] [−auth_level num] [−verify_depth num] [−verify_email email] [−verify_hostname hostname] [−verify_ip ip] [−verify_name name] [−x509_strict]

DESCRIPTION

The **ts** command is a basic Time Stamping Authority (TSA) client and server application as specified in RFC 3161 (Time-Stamp Protocol, TSP). A TSA can be part of a PKI deployment and its role is to provide long term proof of the existence of a certain datum before a particular time. Here is a brief description of the protocol:

1. The TSA client computes a one-way hash value for a data file and sends the hash to the TSA.
2. The TSA attaches the current date and time to the received hash value, signs them and sends the timestamp token back to the client. By creating this token the TSA certifies the existence of the original data file at the time of response generation.
3. The TSA client receives the timestamp token and verifies the signature on it. It also checks if the token contains the same hash value that it had sent to the TSA.

There is one DER encoded protocol data unit defined for transporting a timestamp request to the TSA and one for sending the timestamp response back to the client. The **ts** command has three main functions: creating a timestamp request based on a data file, creating a timestamp response based on a request, verifying if a response corresponds to a particular request or a data file.

There is no support for sending the requests/responses automatically over HTTP or TCP yet as suggested in RFC 3161. The users must send the requests either by ftp or e−mail.

OPTIONS

Time Stamp Request generation

The **−query** switch can be used for creating and printing a timestamp request with the following options:

−rand file...

A file or files containing random data used to seed the random number generator. Multiple files can be specified separated by an OS-dependent character. The separator is**;** for MS-W indows, **,** for OpenVMS, and **:** for all others.

[**−writerand file**]

Writes random data to the specified *file* upon exit. This can be used with a subsequent**−rand** flag.

−config configfile

The configuration file to use. Optional; for a description of the default value, see "COMMAND SUMMARY" in **openssl** (1).

−data file_to_hash
The data file for which the timestamp request needs to be created. stdin is the default if neither the **−data** nor the **−digest** parameter is specified. (Optional)

−digest digest_bytes
It is possible to specify the message imprint explicitly without the data file. The imprint must be specified in a hexadecimal format, two characters per byte, the bytes optionally separated by colons (e.g. 1A:F6:01:... or 1AF601...). The number of bytes must match the message digest algorithm in use. (Optional)

−*digest*
The message digest to apply to the data file. Any digest supported by the OpenSSL **dgst** command can be used. The default is SHA−1. (Optional)

−tspolicy object_id
The policy that the client expects the TSA to use for creating the timestamp token. Either the dotted OID notation or OID names defined in the config file can be used. If no policy is requested the TSA will use its own default policy. (Optional)

−no_nonce
No nonce is specified in the request if this option is given. Otherwise a 64 bit long pseudo-random none is included in the request. It is recommended to use nonce to protect against replay-attacks. (Optional)

−cert
The TSA is expected to include its signing certificate in the response. (Optional)

−in request.tsq
This option specifies a previously created timestamp request in DER format that will be printed into the output file. Useful when you need to examine the content of a request in human-readable format. (Optional)

−out request.tsq
Name of the output file to which the request will be written. Default is stdout. (Optional)

−text
If this option is specified the output is human-readable text format instead of DER. (Optional)

Time Stamp Response generation

A timestamp response (TimeStampResp) consists of a response status and the timestamp token itself (ContentInfo), if the token generation was successful. The **−reply** command is for creating a timestamp response or timestamp token based on a request and printing the response/token in human-readable format. If **−token_out** is not specified the output is always a timestamp response (TimeStampResp), otherwise it is a timestamp token (ContentInfo).

−config configfile
The configuration file to use. Optional; for a description of the default value, see "COMMAND SUMMARY" in **openssl**(1). See **CONFIGURATION FILE OPTIONS** for configurable variables.

−section tsa_section
The name of the config file section containing the settings for the response generation. If not specified the default TSA section is used, see **CONFIGURATION FILE OPTIONS** for details. (Optional)

−queryfile request.tsq
The name of the file containing a DER encoded timestamp request. (Optional)

−passin password_src
Specifies the password source for the private key of the TSA. See "Pass Phrase Options" in **openssl**(1). (Optional)

−signer tsa_cert.pem
The signer certificate of the TSA in PEM format. The TSA signing certificate must have exactly one extended key usage assigned to it: timeStamping. The extended key usage must also be critical,

otherwise the certificate is going to be refused. Overrides the **signer_cert** variable of the config file. (Optional)

−inkey file_or_id
The signer private key of the TSA in PEM format. Overrides the **signer_key** config file option. (Optional) If no engine is used, the argument is taken as a file; if an engine is specified, the argument is given to the engine as a key identifier.

−*digest*
Signing digest to use. Overrides the **signer_digest** config file option. (Mandatory unless specified in the config file)

−chain certs_file.pem
The collection of certificates in PEM format that will all be included in the response in addition to the signer certificate if the **−cert** option was used for the request. This file is supposed to contain the certificate chain for the signer certificate from its issuer upwards. The **−reply** command does not build a certificate chain automatically. (Optional)

−tspolicy object_id
The default policy to use for the response unless the client explicitly requires a particular TSA policy. The OID can be specified either in dotted notation or with its name. Overrides the **default_policy** config file option. (Optional)

−in response.tsr
Specifies a previously created timestamp response or timestamp token (if **−token_in** is also specified) in DER format that will be written to the output file. This option does not require a request, it is useful e.g. when you need to examine the content of a response or token or you want to extract the timestamp token from a response. If the input is a token and the output is a timestamp response a default 'granted' status info is added to the token. (Optional)

−token_in
This flag can be used together with the **−in** option and indicates that the input is a DER encoded timestamp token (ContentInfo) instead of a timestamp response (TimeStampResp). (Optional)

−out response.tsr
The response is written to this file. The format and content of the file depends on other options (see **−text**, **−token_out**). The default is stdout. (Optional)

−token_out
The output is a timestamp token (ContentInfo) instead of timestamp response (TimeStampResp). (Optional)

−text
If this option is specified the output is human-readable text format instead of DER. (Optional)

−engine id
Specifying an engine (by its unique **id** string) will cause **ts** to attempt to obtain a functional reference to the specified engine, thus initialising it if needed. The engine will then be set as the default for all available algorithms. Default is builtin. (Optional)

Time Stamp Response verification

The **−verify** command is for verifying if a timestamp response or timestamp token is valid and matches a particular timestamp request or data file. The **−verify** command does not use the configuration file.

−data file_to_hash
The response or token must be verified against file_to_hash. The file is hashed with the message digest algorithm specified in the token. The**−digest** and **−queryfile** options must not be specified with this one. (Optional)

−digest digest_bytes
The response or token must be verified against the message digest specified with this option. The number of bytes must match the message digest algorithm specified in the token. The **−data** and

−queryfile options must not be specified with this one. (Optional)

−queryfile request.tsq

The original timestamp request in DER format. The **−data** and **−digest** options must not be specified with this one. (Optional)

−in response.tsr

The timestamp response that needs to be verified in DER format. (Mandatory)

−token_in

This flag can be used together with the **−in** option and indicates that the input is a DER encoded timestamp token (ContentInfo) instead of a timestamp response (TimeStampResp). (Optional)

−CApath trusted_cert_path

The name of the directory containing the trusted CA certificates of the client. See the similar option of **verify** (1) for additional details. Either this option or **−CAfile** must be specified. (Optional)

−CAfile trusted_certs.pem

The name of the file containing a set of trusted self-signed CA certificates in PEM format. See the similar option of **verify** (1) for additional details. Either this option or **−CApath** must be specified. (Optional)

−untrusted cert_file.pem

Set of additional untrusted certificates in PEM format which may be needed when building the certificate chain for the TSA's signing certificate. This file must contain the TSA signing certificate and all intermediate CA certificates unless the response includes them. (Optional)

verify options

The options **−attime timestamp**, **−check_ss_sig**, **−crl_check**, **−crl_check_all**, **−explicit_policy**, **−extended_crl**, **−ignore_critical**, **−inhibit_any**, **−inhibit_map**, **−issuer_checks**, **−no_alt_chains**, **−no_check_time**, **−partial_chain**, **−policy**, **−policy_check**, **−policy_print**, **−purpose**, **−suiteB_128**, **−suiteB_128_only**, **−suiteB_192**, **−trusted_first**, **−use_deltas**, **−auth_level**, **−verify_depth**, **−verify_email**, **−verify_hostname**, **−verify_ip**, **−verify_name**, and **−x509_strict** can be used to control timestamp verification. See**v erify** (1).

CONFIGURATION FILE OPTIONS

The **−query** and **−reply** commands make use of a configuration file. See **config** (5) for a general description of the syntax of the config file. The **−query** command uses only the symbolic OID names section and it can work without it. However, the **−reply** command needs the config file for its operation.

When there is a command line switch equivalent of a variable the switch always overrides the settings in the config file.

tsa section, **default_tsa**

This is the main section and it specifies the name of another section that contains all the options for the **−reply** command. This default section can be overridden with the **−section** command line switch. (Optional)

oid_file

See **ca** (1) for description. (Optional)

oid_section

See **ca** (1) for description. (Optional)

RANDFILE

See **ca** (1) for description. (Optional)

serial

The name of the file containing the hexadecimal serial number of the last timestamp response created. This number is incremented by 1 for each response. If the file does not exist at the time of response generation a new file is created with serial number 1. (Mandatory)

crypto_device

Specifies the OpenSSL engine that will be set as the default for all available algorithms. The default value is builtin, you can specify any other engines supported by OpenSSL (e.g. use chil for the NCipher HSM). (Optional)

signer_cert

TSA signing certificate in PEM format. The same as the **−signer** command line option. (Optional)

certs

A file containing a set of PEM encoded certificates that need to be included in the response. The same as the **−chain** command line option. (Optional)

signer_key

The private key of the TSA in PEM format. The same as the **−inkey** command line option. (Optional)

signer_digest

Signing digest to use. The same as the ***−digest*** command line option. (Mandatory unless specified on the command line)

default_policy

The default policy to use when the request does not mandate any policy. The same as the **−tspolicy** command line option. (Optional)

other_policies

Comma separated list of policies that are also acceptable by the TSA and used only if the request explicitly specifies one of them. (Optional)

digests

The list of message digest algorithms that the TSA accepts. At least one algorithm must be specified. (Mandatory)

accuracy

The accuracy of the time source of the TSA in seconds, milliseconds and microseconds. E.g. secs:1, millisecs:500, microsecs:100. If any of the components is missing zero is assumed for that field. (Optional)

clock_precision_digits

Specifies the maximum number of digits, which represent the fraction of seconds, that need to be included in the time field. The trailing zeros must be removed from the time, so there might actually be fewer digits, or no fraction of seconds at all. Supported only on UNIX platforms. The maximum value is 6, default is 0. (Optional)

ordering

If this option is yes the responses generated by this TSA can always be ordered, even if the time difference between two responses is less than the sum of their accuracies. Default is no. (Optional)

tsa_name

Set this option to yes if the subject name of the TSA must be included in the TSA name field of the response. Default is no. (Optional)

ess_cert_id_chain

The SignedData objects created by the TSA always contain the certificate identifier of the signing certificate in a signed attribute (see RFC 2634, Enhanced Security Services). If this option is set to yes and either the **certs** variable or the **−chain** option is specified then the certificate identifiers of the chain will also be included in the SigningCertificate signed attribute. If this variable is set to no, only the signing certificate identifier is included. Default is no. (Optional)

ess_cert_id_alg

This option specifies the hash function to be used to calculate the TSA's public key certificate identifier. Default is sha1. (Optional)

EXAMPLES

All the examples below presume that **OPENSSL_CONF** is set to a proper configuration file, e.g. the example configuration file openssl/apps/openssl.cnf will do.

Time Stamp Request

To create a timestamp request for design1.txt with SHA−1 without nonce and policy and no certificate is required in the response:

```
openssl ts −query −data design1.txt −no_nonce \
      −out design1.tsq
```

To create a similar timestamp request with specifying the message imprint explicitly:

```
openssl ts −query −digest b7e5d3f93198b38379852f2c04e78d73abdd0f4b \
       −no_nonce −out design1.tsq
```

To print the content of the previous request in human readable format:

```
openssl ts −query −in design1.tsq −text
```

To create a timestamp request which includes the MD−5 digest of design2.txt, requests the signer certificate and nonce, specifies a policy id (assuming the tsa_policy1 name is defined in the OID section of the config file):

```
openssl ts −query −data design2.txt −md5 \
      −tspolicy tsa_policy1 −cert −out design2.tsq
```

Time Stamp Response

Before generating a response a signing certificate must be created for the TSA that contains the **timeStamping** critical extended key usage extension without any other key usage extensions. You can add this line to the user certificate section of the config file to generate a proper certificate;

```
 extendedKeyUsage = critical,timeStamping
```

See **req**(1), **ca**(1), and **x509**(1) for instructions. The examples below assume that cacert.pem contains the certificate of the CA, tsacert.pem is the signing certificate issued by cacert.pem and tsakey.pem is the private key of the TSA.

To create a timestamp response for a request:

```
openssl ts −reply −queryfile design1.tsq −inkey tsakey.pem \
      −signer tsacert.pem −out design1.tsr
```

If you want to use the settings in the config file you could just write:

```
openssl ts −reply −queryfile design1.tsq −out design1.tsr
```

To print a timestamp reply to stdout in human readable format:

```
openssl ts −reply −in design1.tsr −text
```

To create a timestamp token instead of timestamp response:

```
openssl ts −reply −queryfile design1.tsq −out design1_token.der −token_out
```

To print a timestamp token to stdout in human readable format:

```
openssl ts −reply −in design1_token.der −token_in −text −token_out
```

To extract the timestamp token from a response:

```
openssl ts −reply −in design1.tsr −out design1_token.der −token_out
```

To add ’granted’ status info to a timestamp token thereby creating a valid response:

```
openssl ts −reply −in design1_token.der −token_in −out design1.tsr
```

Time Stamp Verification

To verify a timestamp reply against a request:

```
  openssl ts −verify −queryfile design1.tsq −in design1.tsr \
        −CAfile cacert.pem −untrusted tsacert.pem
```

To verify a timestamp reply that includes the certificate chain:

```
  openssl ts −verify −queryfile design2.tsq −in design2.tsr \
        −CAfile cacert.pem
```

To verify a timestamp token against the original data file:

```
 openssl ts −verify −data design2.txt −in design2.tsr \
    −CAfile cacert.pem
```

To verify a timestamp token against a message imprint:

```
 openssl ts −verify −digest b7e5d3f93198b38379852f2c04e78d73abdd0f4b \
    −in design2.tsr −CAfile cacert.pem
```

You could also look at the ’test’ directory for more examples.

BUGS

- No support for timestamps over SMTP, though it is quite easy to implement an automatic e−mail based TSA with **procmail**(1) and **perl**(1). HTTP server support is provided in the form of a separate apache module. HTTP client support is provided by **tsget**(1). Pure TCP/IP protocol is not supported.
- The file containing the last serial number of the TSA is not locked when being read or written. This is a problem if more than one instance of **openssl**(1) is trying to create a timestamp response at the same time. This is not an issue when using the apache server module, it does proper locking.
- Look for the FIXME word in the source files.
- The source code should really be reviewed by somebody else, too.
- More testing is needed, I have done only some basic tests (see test/testtsa).

SEE ALSO

tsget(1), **openssl**(1), **req**(1), **x509**(1), **ca**(1), **genrsa**(1), **config**(5)

COPYRIGHT