NAME

ksi sign - Sign data with KSI command-line tool.

SYNOPSIS

ksi sign [**-o** *out.ksig*]... **-S** *URL* [**--aggr-user** *user* **--aggr-key** *key*] [**-H** *alg*] [**--data-out** *file*] [**--dump**] [*more options*] [**-i** *input*]... [*input*]... [**--** *only_file_input...*]

DESCRIPTION

Signs the given input such as content of a given file or a precomputed hash value with KSI. User must have access to KSI signing service (KSI Aggregator) for signing. Note that until not extended, the signatures can be verified using the copy of calendar blockchain at KSI Extender (calendar-based verification) or using the PKI signature in the calendar authentication record (key-based verification, temporary only). See **ksi-verify**(1) for details.

OPTIONS

-i *input* The input is either the path to the file to be hashed and signed or a hash imprint in the case the data to be signed has been hashed already. Use '-' as file name to read data to be hashed from *stdin*. Hash imprint format: <*alg*>:<*hash in hex*>, where <*alg*> marks the hash algorithm and <*hash in hex*> is hash value in hex format. Use **-h** to get the list of supported hash algorithms. Note that only the main data stream of the file is hashed, any extensions of specific file systems are ignored (e.g. the alternate data streams of NTFS are not signed). Flag **-i** can be omitted when specifying the input. Without **-i** it is not possible to sign files that look like command line parameters (e.g. -a, --option). To interpret all inputs as regular files no matter what the file's name is, see parameter **--**.

-o *out.ksig*

Define the output file's path for the signature. Use '-' as file name to redirect signature binary stream to *stdout*. If not specified, the output is saved to the same directory where the input file is located. If specified as directory, all the signatures are saved there. When signature's output file name is not explicitly specified the signature is saved to <input file>.ksig (or <input file>_<nr>.ksig, where <nr> is auto-incremented counter if the output file already exists). When there are N x input and explicitly specified N x output every signature is saved to the corresponding path. If output file name is explicitly specified, will always overwrite the existing file.

-H *alg* Use the given hash algorithm to hash the file to be signed. Use **ksi -h** to get the list of supported hash algorithms.

-S *URL* Specify the signing service (KSI Aggregator) URL.

--aggr-user *user*

Specify the username for signing service.

--aggr-key *key*

Specify the HMAC key for signing service.

--data-out *file*

Save signed data to file. Use when signing a stream. Use '-' as file name to redirect data being hashed to *stdout*.

--max-lvl *int*

Set the maximum depth (0 - 255) of the local aggregation tree (default: 0). It must be noted that when using masking (**--mask**) or embedding the metadata (**--mdata**), the maximum count of document hash values that could be signed during a single local aggregation round will be reduced. To enable signing in multiple local aggregation rounds see **--max-aggr-rounds**.

--max-aggr-rounds *int*

Set the upper limit of local aggregation rounds that may be performed (default: 1).

--mask [<*hex* | *alg:[arg...]*>]

Specify a hex string to initialize and apply the masking process, or specify an algorithm to generate the initial value instead. See **--prev-leaf** to see how to link another aggregation tree to current aggregation process. Supported algorithms:

- **crand**:*seed,len* - Use standard C rand() function to generate array of random numbers with the given seed and length. The seed value is unsigned 32bit integer or 'time' to use the system time value instead. If function is specified without the arguments (crand:) 'time' is used to generate random array with size of 32 bytes.

When mask is specified without the argument the default mask 'crand:' is used.

--prev-leaf *hash*
: Specify the hash imprint of the last leaf from another local aggregation tree to link it with the current first local aggregation round. Hash imprint format: <*alg*>:<*hash in hex*>, where <*alg*> marks the hash algorithm and <*hash in hex*> is hash value in hex format. Use **-h** to get the list of supported hash algorithms. Is valid only with option **--mask**.

--mdata
: Embed metadata to the KSI signature. To configure metadata at least **--mdata-cli-id** must be specified. See also other options **--mdata-***.

--mdata-cli-id *str*
: Specify client ID as a string that will be embedded into the signature as metadata. It is mandatory part of the metadata.

--mdata-mac-id *str*
: Specify machine ID as a string that will be embedded into the signature as metadata. It is optional part of metadata.

--mdata-sqn-nr [*int*]
: Specify incremental (sequence number is incremented in every aggregation round) sequence number of the request as integer that will be embedded into the signature as metadata. If the parameter is given without the argument, 0 is used. It is optional part of metadata.

--mdata-req-tm
: Embed request time extracted from the machine clock into the signature as metadata. It is optional part of metadata.

--
: If used, **everything** specified after the token is interpreted as **input file** (command-line parameters (e.g. --conf, -d), *stdin* (-) and precomputed hash imprints (SHA-256:7647c6...) are all interpreted as regular files).

-d
: Print detailed information about processes and errors to *stderr*.

--dump
: Dump signature(s) created in human-readable format to *stdout*.

--show-progress
: Show progress bar and print it to *stderr*. Is only valid with **-d**. It must be noted that progress bar hides some debug information.

--conf *file*
: Read configuration options from given file. It must be noted that configuration options given explicitly on command line will override the ones in the configuration file. See **ksi-conf**(5) for more information.

--log *file*
: Write libksi log to given file. Use '-' as file name to redirect log to *stdout*.

EXIT STATUS

See **ksi**(1) for more information.

EXAMPLES

In the following examples it is assumed that KSI service configuration options (URLs, access credentials) are defined. See **ksi-conf**(5) for more information.

1 To sign a file *file* and save signature to *sig.ksig*:

ksi sign -i *file* **-o** *sig.ksig*

2 To sign a data hash (hashed with SHA256) and save the resulting signature to file *sig.ksig*:

ksi sign -i *SHA-256:c8ef6d57ac28d1b4e95a513959\f5fcdd0688380a43d601a5ace1d2e96884690a* **-o** *sig.ksig*

3 To sign a data file *file* with non-default algorithm *SHA1*:

ksi sign -i *file* **-H** *SHA1* **-o** *sig.ksig*

4 To sign a stream (*stdin*), save data from stream to *file* and save signature to *sig.ksig*:

ksi sign -i - **--data-out** *file* **-o** *sig.ksig*

5 To perform local aggregation on files **.doc* and save all the signatures to directory *doc/sig*:

ksi sign -o *doc/sig* **--max-lvl** *5* **--** **.doc*

6 To embed user ID "*My Name*" as metadata to the signature of document *file*:

ksi sign *file* **--max-lvl** *2* **--mdata --mdata-cli-id** "*My Name*"

7 To sign more files than the user is permitted to sign during a single aggregation request:

ksi sign -o *doc/sig* **--max-lvl** *5* **--max-aggr-rounds 10 --** **.doc*

8 To sign multiple files and enable masking with default configuration:

ksi sign -o *doc/sig* **--max-lvl** *5* **--mask --** **.doc*

ENVIRONMENT

Use the environment variable **KSI_CONF** to define the default configuration file. See **ksi-conf**(5) for more information.

AUTHOR

Guardtime AS, http://www.guardtime.com/

SEE ALSO

ksi(1), **ksi-verify**(1), **ksi-extend**(1), **ksi-pubfile**(1), **ksi-conf**(5)