NAME

ksi verify - Verify KSI signature with KSI command-line tool.

SYNOPSIS

ksi verify -i *in.ksig* [**-f** *data*] [*more_options*]

ksi verify --ver-int -i *in.ksig* [**-f** *data*] [*more_options*]

ksi verify --ver-cal -i *in.ksig* [**-f** *data*] **-X** *URL* [**--ext-user** *user* **--ext-key** *key*] [*more_options*]

ksi verify --ver-key -i *in.ksig* [**-f** *data*] **-P** *URL* [**--cnstr** *oid=value*]... [**-V** *cert.pem*]... [**-W** *dir*]... [*more_options*]

ksi verify --ver-pub -i *in.ksig* [**-f** *data*] **--pub-str** *pubstr* [**-x -X** *URL* [**--ext-user** *user* **--ext-key** *key*]] [*more_options*]

ksi verify --ver-pub -i *in.ksig* [**-f** *data*] **-P** *URL* [**--cnstr** *oid=value*]... [**-V** *cert.pem*]... [**-W** *dir*]... [**-x -X** *URL* [**--ext-user** *user* **--ext-key** *key*]] [*more_options*]

DESCRIPTION

Verifies the given KSI signature and, if given, the file or its precomputed hash value. There are four main verification policies:

- Internal verification (**--ver-int**). Only internal consistency of the signature is checked and no trust anchor is used and no external resources are needed. This check is also performed as the first step in all other policies.
- Calendar-based verification (**--ver-cal**). Signature is verified against calendar blockchain database at the KSI Extender. Verification is done by checking that the output hash value computed from the aggregation hash chain matches the corresponding entry in the calendar blockchain. Access to KSI Extender is needed.
- Key-based verification (**--ver-key**). Signature must contain a calendar hash chain and a calendar authentication record that can be verified against the signing certificates. To be able to perform key-based verification user must have an access to a trusted KSI publications file with signing certificates in it.
- Publication-based verification (**--ver-pub**). Signature must be extended to a time of publication and contain a publication record unless automatic extension of the signature is enabled with **-x**. Verification is done by checking that the publication record in the signature matches a publication in the publications file or the publication string given on the command line. Publications file or publication string retrieved from printed media is needed.

Note that only publication-based verification should be preferred in long term as it does not rely on any keys and trusted services. The other policies can be used temporarily when the signature is created and there is no publication yet to extend the signature to. Default verification policy is used if policy is not explicitly specified. Default policy uses publication-based and key-based verification where the preferable method is selected according to the given verification info. Default policy is affected by command-line options as the other policies.

OPTIONS

--ver-int
: Perform internal verification.

--ver-cal
: Perform calendar-based verification (use extending service).

--ver-key
: Perform key-based verification.

--ver-pub
: Perform publication-based verification (use with **-x** to permit extending).

-i *in.ksig*
: Specify the signature file to be verified. Use '**-**' as file name to read signature file from *stdin*. 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).

-f *data*
: Specify file to be hashed or precomputed data hash imprint to extract the hash value that is going to be verified. Hash format: <alg>:<hash in hex>. Use '-' as file name to read data to be hashed from *stdin*. Call **ksi -h** to get the list of supported hash algorithms.

-X *URL*
: Specify the extending service (KSI Extender) URL.

--ext-user *str*
: Specify the username for extending service.

--ext-key *str*
: Specify the HMAC key for extending service.

--ext-hmac-alg *alg*
: Hash algorithm to be used for computing HMAC on outgoing messages towards KSI extender. If not set, default algorithm is used. Use **ksi -h** to get the list of supported hash algorithms.

-x
: Permit to use extender for publication-based verification.

--pub-str *str*
: Specify the publication string to verify with.

-P *URL*
: Specify the publications file URL (or file with URI scheme 'file://').

--cnstr *oid=value*
: Specify the OID of the PKI certificate field (e.g. e-mail address) and the expected value to qualify the certificate for verification of publications file's PKI signature. At least one constraint must be defined. All values from lower priority source are ignored (see **ksi-conf**(5) for more information).

  For more common OIDs there are convenience names defined:

  - **E** or **email** for OID 1.2.840.113549.1.9.1
  - **CN** or **cname** for OID 2.5.4.3
  - **C or country** for OID 2.5.4.6
  - **O** or **org** for OID 2.5.4.10

-V *file*
: Specify the certificate file in PEM format for publications file verification. All values from lower priority source are ignored (see **ksi-conf**(5)).

-W *dir*
: Specify an OpenSSL-style trust store directory for publications file verification. All values from lower priority source are ignored (see **ksi-conf**(5)).

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

--dump [*G*]
: Dump signature and document hash being verified in human-readable format to *stdout*. In verification report *OK* means that the step is performed successfully, *NA* means that it could not be performed as there was not enough information and *FAILED* means that the verification was unsuccessful. To make signature dump suitable for processing with grep, use '**G**' as argument.

--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. Signature files with extension **.ext.ksig** are extended and files with extension **.ksig** are not.

1 To perform internal verification of the KSI signature *test.ksig* and the data in the file *test*:

ksi verify --ver-int -i *test.ksig* **-f** *test*

2 To perform key-based verification of the KSI signature *test.ksig* and given document hash:

ksi verify --ver-key -i *test.ksig* **-f** *SHA-256:c8ef6d57ac28d1b4e95a513959f5fcdd0688380a43d601a5ace1d2e96884690a*

3 To perform calendar-based verification of the KSI signature *test.ksig*:

ksi verify --ver-cal -i *test.ksig*

4 To perform publication-based verification of the KSI signature *test.ext.ksig*, using publication string:

ksi verify --ver-pub -i *test.ext.ksig* **--pub-str** *AAAAAA-CWYEKQ-AAIYPA-UJ4GRT-HXMFBE-OTB4AB-XH3PT3-KNIKGV-PYCJXU-HL2TN4-RG6SCC-3ZGSBM*

5 To perform publication-based verification of the KSI signature *test.ext.ksig*, using a publications file which is auto-downloaded and verified based on the default configuration options:

ksi verify --ver-pub -i *test.ext.ksig*

6 To perform publication-based verification of the KSI signature *test.ksig*, possibly extending it on the fly:

ksi verify --ver-pub -i *test.ksig* **-x**

7 To perform verification of the KSI signature *test.ksig* using any policy possible, depending on the current state of the signature and dump its content:

ksi verify -i *test.ksig* **--dump**

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