Rocky Enterprise Linux 9.2 Manual Pages on command 'jose-jws-sig.1'

$ man jose-jws-sig.1

JOSE-JWS-SIG(1) JOSE-JWS-SIG(1)

NAME

jose-jws-sig - Signs a payload using one or more JWKs

SYNOPSIS

jose jws sig [-i JWS] [-I PAY] [-s SIG] -k JWK [-o JWS] [-O PAY] [-c]

OVERVIEW

The jose jws sig command signs a payload using one or more JWKs. The payload can be provided either in its decoded form (-I) or embedded in an existing JWS (-i).

A detached JWS can be created by specifying the -O option. In this case, the decoded payload will be written to the output specified and will not be included in the JWS.

If only one key is used (-k), the resulting JWS may be output in JWS Compact Serialization by using the -c option.

This command uses a template based approach for constructing a JWS. You can specify templates of the JWS itself (-i) or for the JWS Signature Object (-r). Attributes specified in either of these templates will appear unmodified in the output. One exception to this rule is that the

JWS Protected Header should be specified in its decoded form in the JWS Signature Object template. This command will automatically encode it as part of the encryption process.

If you specify a JOSE Header Parameter (via either the -i or -r options) that affects the construction of the JWE, this command will attempt to behave according to this parameter as if it were configuration. Currently, jose will modify its behavior for the "alg" JOSE Header Parameter (see RFC 7515 Section 4.1.1).

However, it is not necessary to provide any templates: jose jwe enc will automatically fill in the "alg" parameter by inferring the correct algorithm from the provided input JWKs. Therefore, the -i and -r options should generally be used for providing extended JWE metadata.

It is possible to specify an existing JWS as the JWS template input (-i). This allows the addition of new signatures to an existing JWS.

OPTIONS

- -i JSON, --input=JSON : Parse JWS template from JSON
- -i FILE, --input=FILE : Read JWS template from FILE
- -i -, --input=- : Read JWS template from standard input
- -I FILE, --detached=FILE : Read decoded payload from FILE
- -I -, --detached=- : Read decoded payload from standard input
- -s JSON, --signature=JSON : Parse JWS signature template from JSON
- -s FILE, --signature=FILE : Read JWS signature template from FILE
- -s -, --signature=- : Read JWS signature template standard input
- -k FILE, --key=FILE : Read JWK(Set) from FILE
- -k -, --key=- : Read JWK(Set) from standard input
- -o FILE, --output=FILE : Write JWS to FILE
- -o -, --output=- : Write JWS to stdout (default)
- -O FILE, --detach=FILE : Detach payload and decode to FILE
- -O -, --detach=- : Detach payload and decode to standard output
- -c, --compact : Output JWS using compact serialization

EXAMPLES

Sign data with a symmetric key using JWE JSON Serialization:

```
$ jose jwk gen -i '{"alg":"HS256"}' -o key.jwk
```

```
$ jose jws sig -I msg.txt -k key.jwk -o msg.jws
```

Sign data using detached JWE Compact Serialization:

```
$ jose jws sig -I msg.txt -k key.jwk -O /dev/null -c -o msg.jws
```

Sign with two keys:

```
$ jose jwk gen -i '{"alg":"ES256"}' -o ec.jwk
$ jose jwk gen -i '{"alg":"RS256"}' -o rsa.jwk
$ jose jws sig -I msg.txt -k ec.jwk -k rsa.jwk -o msg.jws
```

AUTHOR

Nathaniel McCallum <npmccallum@redhat.com>

SEE ALSO

jose-jws-sig(1), jose-jws-ver(1)