NAME

Module::Signature - Module signature file manipulation

SYNOPSIS

As a shell command:

```
% cpansign # verify an existing SIGNATURE, or
# make a new one if none exists

% cpansign sign # make signature; overwrites existing one
% cpansign -s # same thing

% cpansign verify # verify a signature
% cpansign -v # same thing
% cpansign -v --skip # ignore files in MANIFEST.SKIP

% cpansign help # display this documentation
% cpansign -h # same thing
```

In programs:

```
use Module::Signature qw(sign verify SIGNATURE_OK);
sign();
sign(overwrite => 1); # overwrites without asking

# see the CONSTANTS section below
(verify() == SIGNATURE_OK) or die "failed!";
```

DESCRIPTION

Module::Signature adds cryptographic authentications to CPAN distributions, via the special *SIGNATURE* file.

If you are a module user, all you have to do is to remember to run `cpansign -v` (or just `cpansign`) before issuing `perl Makefile.PL` or `perl Build.PL`; that will ensure the distribution has not been tampered with.

Module authors can easily add the *SIGNATURE* file to the distribution tarball; see "NOTES" below for how to do it as part of `make dist`.

If you *really* want to sign a distribution manually, simply add `SIGNATURE` to *MANIFEST*, then type `cpansign -s`immediately before `make dist`. Be sure to delete the *SIGNA TURE* file afterwards.

Please also see "NOTES" about *MANIFEST.SKIP* issues, especially if you are using **Module::Build** or writing your own *MANIFEST.SKIP*.

VARIABLES

No package variables are exported by default.

`$Verbose`
: If true, Module::Signature will give information during processing including gpg output. If false, Module::Signature will be as quiet as possible as long as everything is working ok. Defaults to false.

`$SIGNATURE`
: The filename for a distribution's signature file. Defaults to `SIGNATURE`.

`$KeyServer`
: The OpenPGP key server for fetching the author's public key (currently only implemented on `gpg`, not `Crypt::OpenPGP` May be set to a false value to prevent this module from fetching public keys.

`$KeyServerPort`
The OpenPGP key server port, defaults to `11371`.

`$Timeout`
Maximum time to wait to try to establish a link to the key server. Defaults to`3`.

`$AutoKeyRetrieve`
Whether to automatically fetch unknown keys from the key server. Defaults to`1`.

`$Cipher`
The default cipher used by the `Digest` module to make signature files. Defaults to `SHA1`, but may be changed to other ciphers via the `MODULE_SIGNATURE_CIPHER` environment variable if the SHA1 cipher is undesirable for the user.

The cipher specified in the *SIGNATURE* file's first entry will be used to validate its integrity. For `SHA1`, the user needs to have any one of these four modules installed: **Digest::SHA** **Digest::SHA1** **Digest::SHA::PurePerl** or (currently nonexistent) **Digest::SHA1::PurePerl**

`$Preamble`
The explanatory text written to newly generated *SIGNATURE* files before the actual entries.

ENVIRONMENT

Module::Signature honors these environment variables:

MODULE_SIGNATURE_CIPHER
Works like `$Cipher`.

MODULE_SIGNATURE_VERBOSE
Works like `$Verbose`.

MODULE_SIGNATURE_KEYSERVER
Works like `$KeyServer`.

MODULE_SIGNATURE_KEYSERVERPORT
Works like `$KeyServerPort`.

MODULE_SIGNATURE_TIMEOUT
Works like `$Timeout`.

CONSTANTS

These constants are not exported by default.

CANNOT_VERIFY (`0E0`)
Cannot verify the OpenPGP signature, maybe due to the lack of a network connection to the key server, or if neither gnupg nor Crypt::OpenPGP exists on the system.

SIGNATURE_OK (`0`)
Signature successfully verified.

SIGNATURE_MISSING (`-1`)
The *SIGNATURE* file does not exist.

SIGNATURE_MALFORMED (`-2`)
The signature file does not contains a valid OpenPGP message.

SIGNATURE_BAD (`-3`)
Invalid signature detected — it might have been tampered with.

SIGNATURE_MISMATCH (`-4`)
The signature is valid, but files in the distribution have changed since its creation.

MANIFEST_MISMATCH (`-5`)
There are extra files in the current directory not specified by the MANIFEST file.

CIPHER_UNKNOWN (-6)
The cipher used by the signature file is not recognized by the `Digest` and `Digest::*` modules.

NOTES

Signing your module as part of `make dist`

The easiest way is to use **Module::Install**

```
sign; # put this before "WriteAll"
WriteAll;
```

For **ExtUtils::MakeMaker** (version 6.18 or above), you may do this:

```
WriteMakefile(
(MM->can('signature_target') ? (SIGN => 1) : ()),
# ... original arguments ...
);
```

Users of **Module::Build** may do this:

```
Module::Build->new(
(sign => 1),
# ... original arguments ...
)->create_build_script;
```

MANIFEST.SKIP Considerations

(The following section is lifted from Iain Truskett's **Test::Signature** module, under the Perl license. Thanks, Iain!)

It is **imperative** that your *MANIFEST* and *MANIFEST.SKIP* files be accurate and complete. If you are using `ExtUtils::MakeMaker` and you do not have a *MANIFEST.SKIP* file, then don't worry about the rest of this. If you do have a *MANIFEST.SKIP* file, or you use `Module::Build` you must read this.

Since the test is run at `make test` time, the distribution has been made. Thus your *MANIFEST.SKIP* file should have the entries listed below.

If you're using `ExtUtils::MakeMaker` you should have, at least:

```
#defaults
Makefile$
blib/
pm_to_blib
blibdirs
```

These entries are part of the default set provided by `ExtUtils::Manifest` which is ignored if you provide your own *MANIFEST.SKIP* file.

If you are using `Module::Build` you should have two extra entries:

```
Build$
_build/
```

If you don't have the correct entries, `Module::Signature` will complain that you have:

```
==> MISMATCHED content between MANIFEST and distribution files! <==
```

You should note this during normal development testing anyway.

Testing signatures

You may add this code as *t/0-signature.t* in your distribution tree:

```
#!/usr/bin/perl

use strict;
print "1..1\n";
```

```
if (!$ENV{TEST_SIGNATURE}) {
print "ok 1 # skip Set the environment variable",
" TEST_SIGNATURE to enable this test\n";
}
elsif (!-s 'SIGNATURE') {
print "ok 1 # skip No signature file found\n";
}
elsif (!eval { require Module::Signature; 1 }) {
print "ok 1 # skip ",
"Next time around, consider install Module::Signature, ",
"so you can verify the integrity of this distribution.\n";
}
elsif (!eval { require Socket; Socket::inet_aton('pool.sks-keyservers.net') }) {
print "ok 1 # skip ",
"Cannot connect to the keyserver\n";
}
else {
(Module::Signature::verify() == Module::Signature::SIGNATURE_OK())
or print "not ";
print "ok 1 # Valid signature\n";
}

__END__
```

If you are already using **Test::More** for testing, a more straightforward version of *t/0-signature.t* can be found in the **Module::Signature** distribution.

Note that `MANIFEST.SKIP` is considered by default only when `$ENV{TEST_SIGNATURE}` is set to a true value.

Also, if you prefer a more full-fledged testing package, and are willing to inflict the dependency of **Module::Build** on your users, Iain Truskett's **Test::Signature** might be a better choice.

SEE ALSO

Digest, Digest::SHA, Digest::SHA1, Digest::SHA::PurePerl

ExtUtils::Manifest, Crypt::OpenPGP, Test::Signature

Module::Install, ExtUtils::MakeMaker, Module::Build

Dist::Zilla::Plugin::Signature

AUTHORS

Audrey Tang <cpan@audreyt.org>

CC0 1.0 Universal