NAME

Crypt::OpenSSL::Random − OpenSSL/LibreSSL pseudo−random number generator access

SYNOPSIS

```
use Crypt::OpenSSL::Random;

Crypt::OpenSSL::Random::random_seed($good_random_data);
Crypt::OpenSSL::Random::random_egd("/tmp/entropy");
Crypt::OpenSSL::Random::random_status() or
  die "Unable to sufficiently seed the random number generator".

my $ten_good_random_bytes = Crypt::OpenSSL::Random::random_bytes(10);
my $ten_ok_random_bytes = Crypt::OpenSSL::Random::random_pseudo_bytes(10);
```

DESCRIPTION

`Crypt::OpenSSL::Random` provides the ability to seed and query the **OpenSSL** and **LibreSSL** library's pseudo-random number generators.

Note: On **LibreSSL** `random_egd()` is not defined.

EXPORT

None by default.

Static Methods

random_bytes (IV num_bytes)
: This function, returns a specified number of cryptographically strong pseudo-random bytes from the PRNG. If the PRNG has not been seeded with enough randomness to ensure an unpredictable byte sequence, then a false value is returned.

random_pseudo_bytes (IV num_bytes)
: This function, is similar to `random_bytes`, but the resulting sequence of bytes are not necessarily unpredictable. They can be used for non-cryptographic purposes and for certain purposes in cryptographic protocols, but usually not for key generation etc.

random_seed (PV random_bytes_string)
: This function seeds the PRNG with a supplied string of bytes. It returns true if the PRNG has sufficient seeding. Note: calling this function with non-random bytes is of limited value at best!

random_egd (PV egd_string)
: This function seeds the PRNG with data from the specified entropy gathering daemon. Returns the number of bytes read from the daemon on success, or `−1` if not enough bytes were read, or if the connection to the daemon failed.

: `libressl` considers this function insecure, so with libressl this function does not exist.

random_status ()
: This function returns true if the PRNG has sufficient seeding.

BUGS

Because of the internal workings of OpenSSL's random library, the pseudo-random number generator (PRNG) accessed by Crypt::OpenSSL::Random will be different than the one accessed by any other perl module. Hence, to use a module such as Crypt::OpenSSL::Random, you will need to seed the PRNG used there from one used here. This class is still advantageous, however, as it centralizes other methods, such as `random_egd`, in one place.

AUTHOR

Ian Robertson, `iroberts@cpan.com`

Now maintained by Reini Urban, `rurban@cpan.org`

LICENSE

SEE ALSO

perl(1), **rand**(3), **RAND_add**(3), **RAND_egd**(3), **RAND_bytes**(3).